目次

時系列異常検知を（だいたい Rust で）実装してみる ・・・・・・・・・・・・・・・・・ 4
1 はじめに ・・ 4
2 実験データ収集 ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 5
2.1 環境準備 ・・ 5
2.2 データのクエリとエクスポート ・・・・・・・・・・・・・・・・・・・・・・・・・・ 8
3 手法の簡単な説明・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 11
4 実装してみよう ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 13
4.1 プロジェクトの準備 ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 13
4.2 型の定義 ・・ 14
4.3 データの読み込みと整形・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 15
4.4 前処理・・ 15
4.5 Saliency Map の計算・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 16
4.6 スコアの計算・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 18
4.7 異常値判定 ・・ 19
4.8 結果の出力 ・・ 19
5 実験結果 ・・・ 20
6 まとめ ・・・ 22
バイナリ解析実践 - UEFI module ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 24
1 はじめに ・・ 24
2 解析環境 ・・ 24
2.1 解析対象 ・・ 24
2.2 解析ソフト ・・ 25
3 ケーススタディ ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 25
3.1 事前準備 ・・ 25
3.2 エントリポイント周辺の解析・・・・・・・・・・・・・・・・・・・・・・・・・・・ 25
3.3 関数プロトタイプ（呼び出し規約）の設定・・・・・・・・・・・・・・ 26
3.4 EFI_STATUS に対するエラーコードの紐付け・・・・・・・・・・・・・ 28
3.5 エントリポイント周辺の解析（修正版）・・・・・・・・・・・・・・・ 28
3.6 答え合わせ ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 33
4 おわりに ・・・ 34
Re:VIEW で同人誌を作ってみた感想 ・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 36
1 はじめに ・・・ 36

2 urandom の執筆状況 ・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 36
3 Re:VIEW のセットアップ ・・・・・・・・・・・・・・・・・・・・・・・・ 37
3.1 バックエンドを LuaLaTeX に変更 ・・・・・・・・・・・・・・・・・・・ 37
3.2 図・表などの番号に LaTeX のカウンターを利用 ・・・・・・・・・・・ 38
3.3 ソースコードや実行結果に listings を利用・・・・・・・・・・・・・ 40
3.4 何もしない @<nop>命令の追加 ・・・・・・・・・・・・・・・・・・・・ 40
3.5 BibTeX をワークフローに組み込めない ・・・・・・・・・・・・・・・・ 41
4 まとめ ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 41
付録 A 参考文献・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 42
著者紹介 ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 43

時系列異常検知を（だいたい Rust で）実装してみる

1 はじめに

今回は時系列データに対する異常検知を実装してみます。

時系列データとは定期的に観測されたデータを時間の流れに沿って並べたものです。コンピュータという文脈において例を挙げれば、システムメトリクスを収集した後、これを時系列順に並べたものが相当します。もう少し具体的な例にすれば、何秒かに 1 回 CPU 使用率だとかメモリ使用量だとかを測定し、これを測定した時刻と共に並べたものは、時系列データと言えるでしょう。

では、そんな時系列データにおける異常とはなんでしょうか。すぐに思いつくのは測定値の急激な増減です。CPU 使用率であれば、ある瞬間に突然 100%に上がってすぐ元に戻った場合、その 100%に上がった観測点は「異常」と言えます。一方で値が変わらないことが「異常」であることもあります。例えば、あるシステムでは毎日午前 1 時になるとディスクへの読み書きが増大するという周期性が見られたところ、ある 1 日だけそのような動きが見られなかったとします。この場合はむしろ値が動かなかったその 1 日が周期性から外れていて「異常」と言えるでしょう。このように、時系列データにおける異常というのはいくつか考えられます。

この本はコンピュータセキュリティの薄い本ということになっていますが、この時系列データの異常検知は何に役立つのでしょうか。先ほどディスクの読み書き量について一定の周期から外れたものは異常と呼べる、という例を出しましたが、これはシステムのメトリクスを通して何らかの異常な動作を検知する例と言えます。異常な動作というのは当然何らかのバグという可能性もありますが、あるいはシステムに侵入した何者かが大量のファイルを読み込んでいるものかもしれません。つまり、単なるシステムの誤動作ではなく、（管理者の意図しない）人による異常な操作であっても検知できるわけです（逆に言えば単純にメトリクスの異常を検知するだけではそれらを区別できません）。

当然そういった機能を持った監視ソフトウェアはあるでしょうし、あと上の例なら侵入された時点で気付けるようにしておけという話なのですが、あくまで趣味と興味の一環として、今回は時系列データにおける異常検知を自前で実装してみます。今回は [1] で紹介されている、Spectral Residual（SR）法を時系列データに応用した手法を用います。

2 実験データ収集

論文と同様に既存のデータセットを使っても良いのですが、今回は実際のディスクの読み取り量を対象に異常検知を試してみます。ここではそのためのデータ収集について説明します。

2.1 環境準備

先に今回使用した PC の構成を示しておきます（表 1）。

表 1: PC 構成

CPU	Intel Core i5-12600KF (6P+4E/16T)
Memory	32GB
Disk	1TB SSD (WD My Passport)
OS	Kubuntu 24.04.01 LTS (Kernel 6.8.0-49)

記事執筆にあたって環境構築手順を再現するため、別のマシンで改めて実行しています。その環境は表 2 のような構成です。

表 2: 手順再現環境

Board	Raspberry Pi 5
Memory	8GB
Disk	1TB SSD (WD Blue SN580)
OS	Raspberry Pi OS 64bit (bookworm)

この環境はあくまで環境構築手順の再現および手順の妥当性確認のために使用しており、実験のデータは表 1 の環境で収集したものです。

さて、今回はメトリクスの収集とエクスポートのために Prometheus、node exporter、Grafana を使います。これらを全て Docker Compose で立ち上げます。リスト 1 のような `compose.yml` を用意します。

リスト 1: データ収集用 compose.yml

```
services:
  prometheus:
    image: prom/prometheus
    ports:
      - 9090:9090
    volumes:
      - ./prometheus.yml:/etc/prometheus/prometheus.yml:ro
      - prometheus-data:/prometheus
```

バイナリ解析実践 - UEFI module

1 はじめに

ここ十年ほどの間に発売された PC の大半には “UEFI firmware” と呼ばれるソフトウェアが組み込まれている。Firmware は主に OS の起動を担当する「縁の下の力持ち」で、UEFI (Unified Extensible Firmware Interface) はその firmware が実装すべきインターフェースの業界標準を指す。つまり、UEFI firmware とは、UEFI に沿って標準化されたインターフェースを実装する firmware 全般を意味する。

UEFI firmware に脆弱性や悪意あるコードが紛れ込んでいると、OS のセキュリティも危機に晒される。UEFI firmware は PC のモデルによって実装が多少異なるため、都度解析が必要になる。また、一般的な PC の UEFI firmware はビルド済みバイナリの形でしか入手できない。このため、UEFI firmware のバイナリ解析にはセキュリティ研究上の意義がある。

UEFI firmware は、複数のモジュール（UEFI module）から構成される。この記事では、UEFI firmware 解析のための一ステップとして、UEFI module の解析例をケーススタディに沿って紹介する。

2 解析環境

2.1 解析対象

解析対象は CRZEFI（`EFI_Driver_Access` レポジトリの commit `ef0638c2c97833d1e3d07211e1c73bb9cfc0187b`）[1] とする。

CRZEFI は、runtime driver と呼ばれる種類の UEFI module で、OS（Windows）のユーザー空間からカーネル空間へのメモリアクセスや関数呼び出しを可能にする[2]。CRZEFI は専ら自身の PC で使用するためのものであって、勝手に他人の PC で使用できないようになっているが、ある種バックドア的な動作をする UEFI module であり、行う操作には悪意あるコードと近い部分がある。この記事では悪意あるコードの解析

1） https://github.com/TheCruZ/EFI_Driver_Access/tree/ef0638c2c97833d1e3d07211e1c73bb9cfc0187b/CRZEFI

2） https://github.com/TheCruZ/EFI_Driver_Access/blob/ef0638c2c97833d1e3d07211e1c73bb9cfc0187b/README.md?plain=1#L2 , https://github.com/TheCruZ/EFI_Driver_Access/blob/ef0638c2c97833d1e3d07211e1c73bb9cfc0187b/README.md?plain=1#L7

を想定し、学習のための練習台として CRZEFI の解析例を示す。

なお、CRZEFI はソースコードが公開されているが、本記事ではまずソースコードを見ないで、ビルドした状態のバイナリのみを解析する。一通り解析を行った後、ソースコードと突き合わせて解析結果を確認する。

2.2 解析ソフト

解析ソフトは次のものを使用する。

逆アセンブラ・逆コンパイラ

IDA Pro 8.3 + x86-64 decompiler [3)]

その他

efiXplorer 6.0 [4)]

ただし、この記事での解析作業は基本的に IDA Free [5)]でも再現できることを確認している。

3 ケーススタディ

3.1 事前準備

CRZEFI をビルド手順[6)]に従ってビルドし、UEFI module のファイルとして `memory.efi` を得る。

3.2 エントリポイント周辺の解析

IDA へ `memory.efi` を読み込み、「Edit > Plugins > efiXplorer」を選択して EFI 向けの解析を適用する。そのままの状態でエントリポイントを逆コンパイルすると次のようになっている。

```
1 EFI_STATUS __fastcall ModuleEntryPoint(
2         EFI_HANDLE ImageHandle,
3         EFI_SYSTEM_TABLE *SystemTable)
4 {
5   sub_A0E0();
6   return sub_3ACC();
7 }
```

ここでは `sub_A0E0` と `sub_3ACC` の 2 関数が呼ばれている。誌面の都合上省略するが、それぞれの関数の内容を確認すると、関数の大きさや複雑さから処理の本体は `sub_3ACC` の方であると推測できる。

3) https://hex-rays.com/ida-pro
4) https://github.com/binarly-io/efiXplorer
5) https://hex-rays.com/ida-free
6) https://github.com/TheCruZ/EFI_Driver_Access/blob/ef0638c2c97833d1e3d07211e1c73bb9cfc0187b/README.md?plain=1#L19

Re:VIEW で同人誌を作ってみた感想

1 はじめに

urandom では、これまで著者によって Markdown と LaTeX の原稿が存在したため、Markdown を Pandoc[1] で LaTeX へ変換していました。この仕組みには次のような 2 つの問題があります。

まず、この Markdown から LaTeX への変換ワークフローは Makefile や、さらに Pandocfilter と呼ばれる Pandoc が色々なマークアップを抽象化した抽象構文木を処理する Python スクリプトを駆使しており、この仕組みを保守し続ける必要があります。次に同人誌を作るときに Markdown では表現力に不足があり、たとえば画像や表にキャプションをつけたい場合や、画像を並べて配置する場合にも Markdown を他と非互換な拡張を利用する必要が生じます。

このような背景で今回は Re:VIEW[2] を利用して本誌を編集してみました。Re:VIEW は独自のマークアップを利用することで EPUB や HTML、そして PDF に出力できる出版システムです。同人誌界隈では TechBooster さんが Re:VIEW のテンプレート[3] を公開しています。

この記事では、本誌の執筆状況をまず説明し、次に Re:VIEW を利用する際に行ったセッティングのコツや困難な部分を解説します。

2 urandom の執筆状況

本誌は Re:VIEW で処理されていますが、記事のソースは 3 つのマークアップが混在しています。

秋弦めい Re:VIEW のマークアップを利用して執筆
op Markdown 執筆し、Re:VIEW 公式の方法で Re:VIEW マークアップへ変換
yyu Re:VIEW ファイルの冒頭で `//embed[latex]` し実質的に LaTeX で執筆

なぜこのように複数のマークアップを用いているかというと、著者によって組版をどの程度がんばりたいか？に差があるからです。LaTeX という難解で複雑で歴史的な

1) https://github.com/jgm/pandoc
2) https://github.com/kmuto/review
3) https://github.com/TechBooster/ReVIEW-Template

ものを許容してでも組版上の表現力を最大限に使いたいか、あるいは Markdown の表現力でよしとするのか、さらに LaTeX・Markdown の中間程度の表現力と組版の自由度な Re:VIEW を使うかといった選択が分かれています。

また、別の理由としてドキュメントのポータビリティーがあります。本誌は A5 のサイズとなっていますが、このサイズを前提にたとえばソースコードに改行をいれて読みやすくする、図のサイズや二段組みにして並べるといった調整を行えば確かに A5 紙面上は美しくなります。一方でそれによって、表示画面サイズが一定ではなく、かつ改ページといった概念が希薄な Web ページや EPUB ではかえって見た目の印象が悪くなるかもしれません。

組版上の表現力や自由度を行使することによって、紙サイズや改ページがあるかないかといった様々な条件が束縛されていきます。逆に組版上の表現力が低いということは、紙サイズといった条件もまた弱くできることを意味します。

この 3 つのマークアップの中で、LaTeX は PDF のあらゆる情報にタッチできるため、一番自由な表現ができますが、それによって最もポータビリティーを失っていると考えることもできます。一方で Markdown は図を並べて配置することすらできませんが、色々なデバイスや形式で最低限見ることはできるようにレンダリングできます。

このように紙での出版のみを考慮してワンオフなドキュメントを作るのか、それとも電子版への掲載の可能性を考慮してポータビリティーを残しておくのかという考え方の違いもあります。

ここで重要なのは、どの考え方が正しいということではなく、Re:VIEW ではこのように 3 つの異なる考え方の著者が 1 つの同人誌を完成させることができるという点です。

3 Re:VIEW のセットアップ

Re:VIEW は最初の状態である程度そのままで PDF を作れるようになっています。ただ、ここでは筆者の好みや urandom の執筆事情を考慮して次のような変更を行っています。

3.1 バックエンドを LuaLaTeX に変更

urandom の同人誌ではこれまでバックエンドに LuaLaTeX を使っていました。しかし、Re:VIEW のデフォルト設定では upLaTeX と `dvipdfmx` を使って PDF を生成します。upLaTeX は LaTeX（TeX） の処理系を改造して日本語対応させたものですが、次のような事情があります。

- upLaTeX は日本語 TeX 開発コミュニティ[4]によって保守されていますが、処理系そ

4） `https://www.texjp.org/`

著者紹介

第 1 章 秋弦めい / mayth

都内某所にて独立傭兵兼ドクター兼プロデューサー兼年上の弟兼指揮官（母港）兼指揮官（エルモ号）をしつつ副業としてソフトウェアエンジニアめいた仕事をしている。

今更ながら iPad Pro (M4) が学マスと同時に出たのは奇跡だと思う。

第 2 章 op

アイマスとにじさんじと初音ミクと CTF とコンピューターセキュリティを行ったり来たりしている。

体の数が足りていない。

第 3 章 yyu

最近、コーヒーインストラクター 2 級になりました。

urandom vol.13

2024 年 12 月 30 日　コミックマーケット 105 版
著　者　urandom
連絡先　https://urandom.team
印刷所　株式会社ポプルス