Rubyをセキュアな言語にする
～Rubyの脆弱性対応の最新動向紹介～

クックパッド株式会社
遠藤侑介 (@mametter)

RubyWorld Conference 2022
(2022/11/10)

遠藤侑介（@mametter）

- クックパッドで働くフルタイムRubyコミッタ
- 「Rubyを堅牢にする」
  - 機能開発面：コードカバレッジ、型解析など
  - 脆弱性対応とセキュリティリリース

脆弱性とは

- 情報セキュリティにかかわる重大なバグのこと
- 放置したら最悪の場合……
  - サービス乗っ取り
  - 機密情報流出
- Rubyのパッチリリースで修正している

本日話すこと

cookpad

- Rubyのセキュリティリリースの裏側
- ReDoS: 近年注目されている脆弱性と、Rubyの対策
  - 注：Ruby開発者の総意ではなく、遠藤の個人的見解も含まれます

Rubyのセキュリティリリースの裏側

セキュリティリリースまでの流れ

cookpad

1. 脆弱性報告を受けとる
2. 脆弱性かどうかを判断する
3. 修正を作成する
4. CVE番号を取得する
5. アナウンス文を準備する
6. リリースする

1. 脆弱性報告を受け取る

cookpad

- HackerOne
  - 脆弱性情報のプラットフォーム
    - OSSやWebサービスの脆弱性を開発者に報告できる
  - Rubyの脆弱性には報奨金が出る
    - The Internet Bug Bounty

https://www.ruby-lang.org/ja/security/

- security@ruby-lang.orgにメールでも可

2. 脆弱性かどうかを判断する

cookpad

- 原則：次の2つがYesならRubyの脆弱性
  - 1. 『現実的にセキュリティ被害が生じる可能性がある』
  - 2. 『Rubyで直すべき問題である』
- 判断が悩ましい場合が非常に多い

脆弱性判断の難しさ

cookpad

- 例:『system関数に信頼できない入力を与えると危険』
  - それをしたコードの責任、Rubyの脆弱性ではない
- 問題となる機能によって判断が変わる
  - 「この機能をそんな風に使うアプリが実在するのか？」
  - 「仮に実在したとしても、アプリ側の責任ではないか？」
- 総合的に判断する
  - 機能のドキュメント、世間の認知、利用状況を調べる
  - 攻撃の容易さ、影響の大きさ、他言語の判断なども加味する

脆弱性判断のFAQ

cookpad

- Q. 少しでも可能性があれば脆弱性と判断すべきでは？
- A. セキュリティリリースのコスト・リスクがあります
  - セキュリティリリース自体のコスト（後述）
  - ユーザの対応コスト（厳しいアップデートポリシーを持つ組織も）
  - 脆弱性の「修正」が非互換となりうる
  - Rubyの判断は他プロジェクトにも影響する
  - （とはいえ、基本的には安全側に倒して判断します）

3. 修正を作成する

cookpad

- 限られたRuby開発者だけでパッチを完成させる

- 現代的なOSS開発のエコシステムが使用できない
  - 公開のGitHubプルリクエストにできない
  - 公開でのレビューやプレビューリリースもできない
  - CIでの検証もリリース直前までできない（これは改善の余地あり）

4. CVE番号を取得する

- 脆弱性情報の識別子
  - Mitre社などで取得できる

- まめ知識：CVE番号があっても脆弱性とは限らない
  - CVE番号は、誰でも取得できる
  - 開発者や専門家による検証プロセスなどはない

5. アナウンス文を準備する

cookpad

- CVE番号
- 脆弱性の詳細
- 影響を受けるバージョン
- クレジット
  - アナウンス文も秘匿管理する

Ruby
A PROGRAMMER'S BEST FRIEND

Google™カスタム検索 検索

ダウンロード ドキュメント ライブラリ コミュニティ コア開発 ニュース セキュリティ Rubyとは

CVE-2022-28738: Regexp コンパイル時のダブルフリー

Posted by mame on 12 Apr 2022
Translated by jinroq

Regexp コンパイル時に、ダブルフリーをする脆弱性が発見されました。 この脆弱性は、CVE-2022-28738 として登録されています。 Ruby をアップグレードすることを強く推奨します。

詳細

Regexp のコンパイル処理にバグがあり、細工したソース文字列で Regexp オブジェクトを作成すると、同じメモリが二度解放される可能性があります。これは「ダブルフリー」と呼ばれる脆弱性です。 一般的に、信頼できない入力から生成された Regexp オブジェクトを作成し、使用することは安全ではないと考えられています。しかしながら、今回のケースでは総合的に判断した結果、この問題を脆弱性として扱うことにしました。

Ruby を 3.0.4 または 3.1.2 に更新してください。

影響を受けるバージョン

- ruby 3.0.3 以前
- ruby 3.1.1 以前

なお、ruby 2.6 系列、2.7 系列は影響を受けません。

クレジット

この脆弱性情報は、piao 氏によって報告されました。

更新履歴

- 2022-04-12 21:00:00 (JST) 初版

最近のニュース

Ruby 3.1.2 リリース
Ruby 3.0.4 リリース
Ruby 2.7.6 リリース
Ruby 2.6.10 リリース
CVE-2022-28738: Regexp コンパイル時のダブルフリー

Syndicate

最近のニュース (RSS)

6.リリースする（準備）

cookpad

- リリース関係者のスケジュールを調整する
  - ブランチメンテナ（nagachika, unak, naruse）
  - パッケージ作成（znz）
  - サーバ管理（hsbt）
  - 問題の機能のメンテナ and/or パッチ作成者（nobuが多い）
  - 当日見守る人（自分）

  - ユーザが対応しやすい平日の火〜木曜日が望ましい
- 関係者にリリース予告する

6.リリースする（当日）

- 残りのすべて
  - パッチを各ブランチにコミットする
  - 各ブランチのCIを確認する（失敗したら対応検討）
  - Rubyのパッケージを作成する
  - パッケージのCIを確認する（失敗したら対応検討）
  - パッケージとアナウンス文を公開する

これまでのプロセス改善

- 他OSSのセキュリティ対応プロセスの研究
  - セキュリティポリシーの調査
    - Perl, Python, PHP, Node.js, Go, Rust
  - 実務者へのヒアリング・情報交換
- 作業の整理
- アナウンス文のGitHub Security Advisoriesによる管理
- チケット管理やCVE取得の半自動化
- パッケージ生成の自動化（by @znz）

これからやりたい改善・個人的野望

- パッチリリースの定期化
- パッチやアナウンス文の秘匿管理のさらなる効率化
- 秘匿したパッチのCIテスト実行
- 脆弱性判断基準の整理と明文化
- セキュリティの専門家によるレビュー体制

お願い

- 現状、作業の多くが無償ボランティアです
  - Ruby Associationによる「Ruby安定版の保守事業」は、バックポート作業以外は現状のスコープではない
  - 業務時間に作業しているコミッタもいるが、主業務ではない
  - 通常のOSS開発と違い、有志の参加を気楽には受けられない
- 継続的なセキュリティリリースと更なる改善のために、Ruby Associationへの寄付をご検討ください
  - https://www.ruby.or.jp/ja/sponsors/

ReDoS: 近年注目されている脆弱性とRubyの対策

話の流れ

- ReDoSとは
- ユーザが現在とれる対策・緩和策
- Rubyが現在検討している根本対策

ReDoSとは

cookpad

- 正規表現（Regular Expression）
  - 簡単に文字列を分解する機能
  - まれに分解に長時間がかかる

```
str = "foo = bar"

str =~ /^(.+)¥s*=¥s*(.+)$/

p $1  #=> "foo"
p $2  #=> "bar"
```

- ReDoS（Regular Expression Denial of Service）
  - 分解に長時間がかかる変な文字列を送り込み、
  - 本来のサービスに使う計算時間を枯渇させる攻撃

  - Rubyに限らず、JavaScriptやPythonなど多言語で流行中

ReDoSの影響

- インパクト：サービスが停止する
  - 情報漏洩や改ざんほどの重大性はないが、
  - 攻撃の容易さや文脈次第でセキュリティ問題になる
- 有名な事例：インターネットが27分間壊れた
  - 『2019年7月2日に発生したCloudflareの停止に関する詳細』
    https://blog.cloudflare.com/ja-jp/details-of-the-cloudflare-outage-on-july-2-2019-ja-jp/

ユーザが今できるReDoS対策

cookpad

- ReDoSを起こす正規表現を修正する、使わない
  - 正規表現の検証には recheck [1] などのツールが有用
- 長過ぎる文字列を正規表現で分解しない
- リクエスト単位でのタイムアウトを設定する
- サービスの状態を監視し、攻撃リクエストを拒絶する
  - ReDoS以外のDoSや、その他の攻撃に対しても有用

[1] https://makenowjust-labs.github.io/recheck/

Rubyが現在検討している
ReDoS根本対策

根本的なReDoS対策を研究中

cookpad

- 問題意識：「正規表現を修正する」をやめたい
  - 修正によってはコードの可読性が下がることもある
  - セキュリティリリースになるとコスト・リスクが高い
- Ruby本体で取りうる根本的な対策
  1. 効率的なアルゴリズムの併用
  2. 組み込みタイムアウトの導入
  3. 既存アルゴリズムの改良

1. 効率的なアルゴリズムの併用（1）

- ReDoSに強い正規表現エンジンを利用する
  - onigmo: 現在Rubyが利用している、ReDoSが起きうる
  - RE2: golangの主要開発者が作成、ReDoSが（ほぼ）起きない
- 問題：onigmoとRE2には多数の非互換がある
- TruffleRubyの解決[2]：両エンジンを併用する
  - なるべくRE2を使うが、無理なときのみonigmoを使う
  - ReDoSの可能性を大きく減らせる

[2] Benoit Daloze and Josef Haider. "Just-in-Time Compiling Ruby Regexps on TruffleRuby".
※TruffleRubyはRE2 + onigmoではなくT-Regex + joniを利用

1.効率的なアルゴリズムの併用（2）

- RubyでもRE2とonigmoを併用できる？ → 厳しかった
  - RE2はUTF-8とASCII文字列しか扱えない
  - 基本機能の中でも非互換が予想以上にあった

```
"abc¥n" =~ /^$/  # RE2ではマッチ、onigmoでは非マッチ
```

    - 試作評価では、約半数の正規表現がonigmoに縮退した

- 本気でやるならRE2を改造する必要がある
  - 実装難度が急激に上がるので、現在は別の対策を検討中

2. 組み込みタイムアウトの導入

cookpad

- 文字列の分解に指定時間以上かかったら止める
  - 小さい非互換で
    多くのReDoSを防ぐ

```
Regexp.timeout = 0.1
str = "…攻撃的な文字列…"
str =~ /¥s*=¥s*/  # 0.1秒で例外を投げる
```

- 注意点
  - 「誤爆」の可能性がある、ユーザの設定が必要
  - 分解を何度も行う場合は防げない
- Ruby 3.2で実験的に導入予定

3. 既存アルゴリズムの改良

cookpad

- onigmoを少し改良し、最悪時性能を大幅に向上する[3]
  - 非互換ゼロで多くのReDoSを防ぐ
- 注意点
  - 扱えない正規表現も少しある
  - 分解時のメモリの使用量は増える
- 現在開発中、乞うご期待 → 昨日マージしました！
  - 藤浪大弥さん（recheck作者）がクックパッドのインターンで開発中

[3] James C.Davis et al.
"Using Selective Memoization to Defeat Regular Expression Denial of Service (ReDoS)"

抜本的な対策案のまとめ

	実装	互換性	対策としての強力さ
1. 効率的なアルゴリズムの併用	中	△ 低い	△ 防げないReDoSが多い
2. 組み込みタイムアウトの導入	易	◯ 高い	◯ 多くのReDoSを防ぐ
3. 既存アルゴリズムの改良	難	◎ 最高	◯ 多くのReDoSを防ぐ

- 次バージョンのRuby 3.2では実験的に (2) を導入予定
- (3) も現在鋭意開発中 → 昨日マージしました！

まとめ

まとめ

cookpad

- Rubyの脆弱性対応プロセスを紹介しました
  - Rubyをセキュアに保つための隠れた努力が行われています
    - 脆弱性判断のための研究や検討
    - 現代的なOSS開発エコシステムが利用できない制約
  - プロセス改善のための支援をお願いします
    - https://www.ruby.or.jp/ja/sponsors/
- ReDoSという脆弱性の紹介と対策を紹介しました
  - 現状では、個別修正や緩和策の併用で回避してください
  - 根本的な解決も検討中なので、今後の改善にご期待ください