CVSS 9.8を含むCVE4件、Fluentdメンテナが経験した脆弱性対応の舞台裏

CloudNative Days Winter 2026

2026/11/19



自己紹介



Shizuo Fujita (藤田 静男)

株式会社クリアコード
Fluentd コミッター

去年は、裏番組に全部持っていかれた

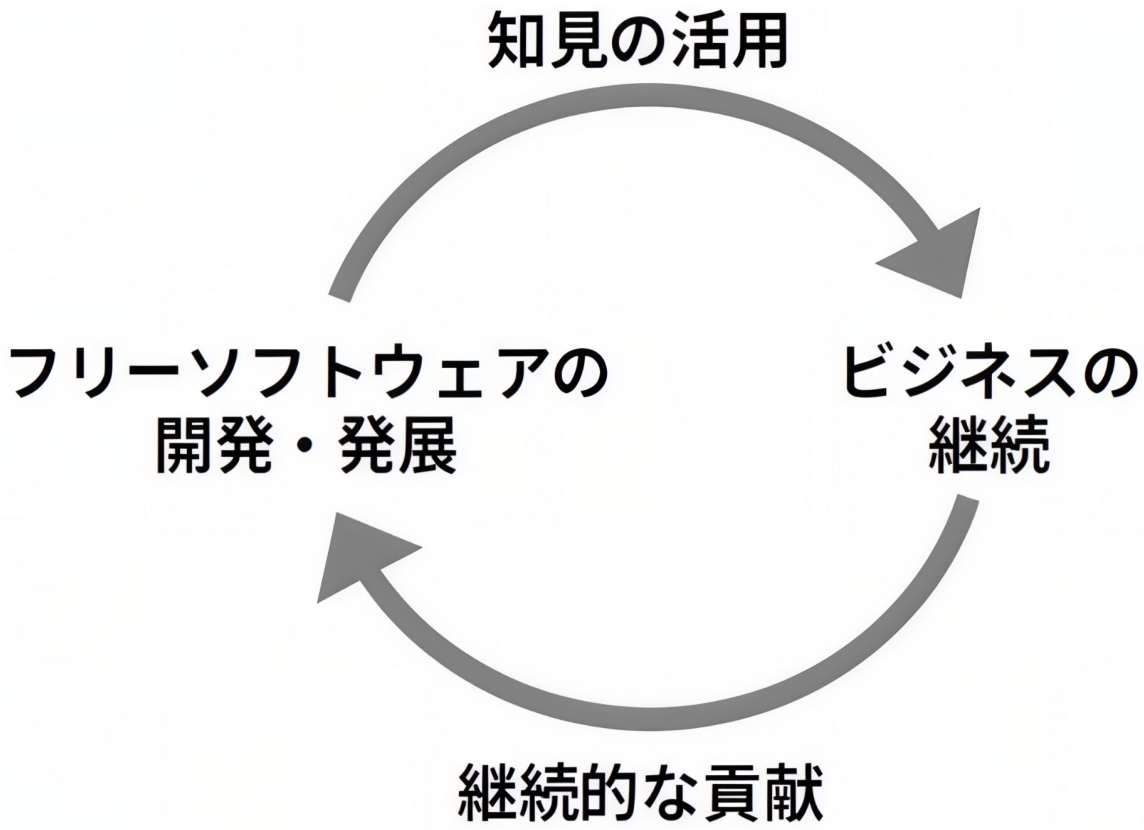
CNDW2025、同じ時間帯のセッション

『ゼルダの伝説 ティアーズ オブ ザ キングダム』の開発を支えたクラウド環境

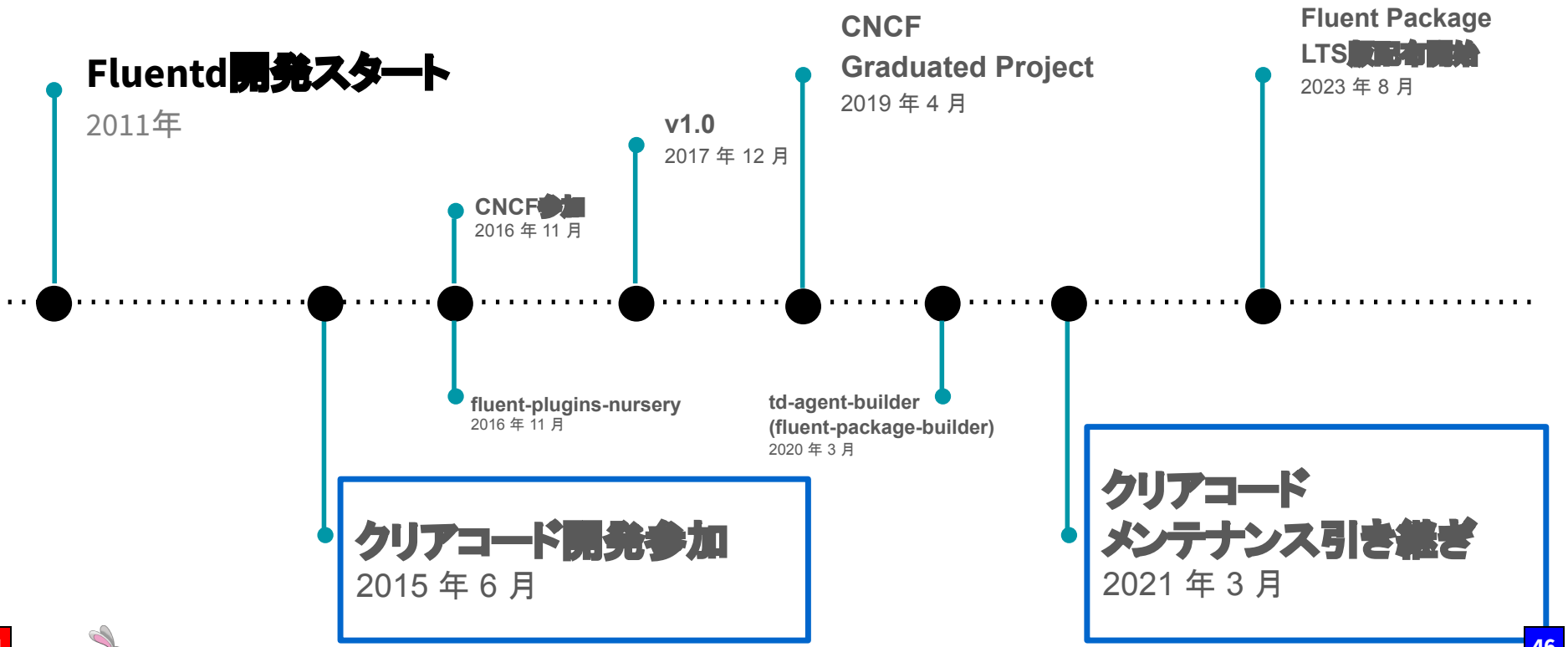
今日はこちらに来てくれて、ありがとうございます



株式会社クリアコード



Fluentdの歴史とクリアコードの関わり



FluentdのCVE4件に、

いつ気づきましたか？



見ている場所で、気づく日が変わる

早い　　遅い

6/26	6/29	7/8	7/13	11/19
当日	3日後	12日後	17日後	今日
GitHubアドバイザリ リリース告知	JVN （日本語）	cve.org / NVD に掲載	NVDで バージョン照合	この発表

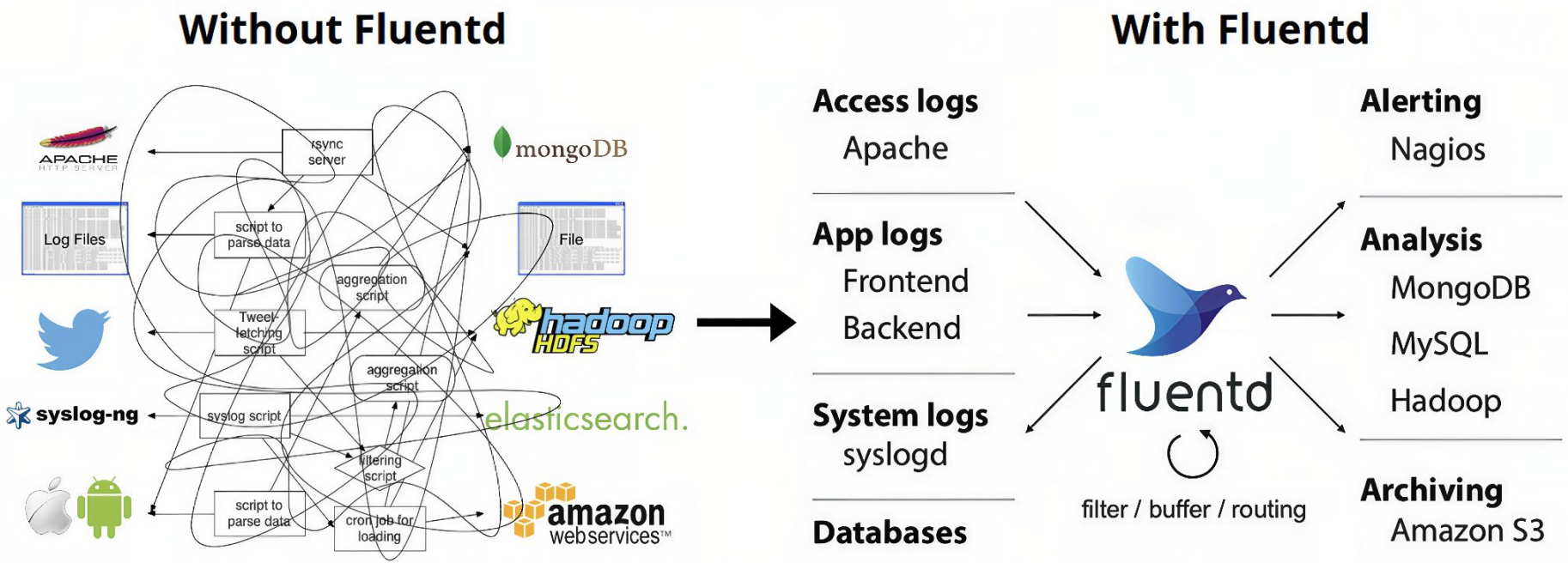
v1.19.4の4件はCVEなし。CVEだけでは、今日も気づけない

今日お話すること

- 前提: Fluentdとその攻撃面
- 何が起きたか: 4件の脆弱性
- 舞台裏: 報告から公開、DB反映まで
- 設定診断ツールを作った判断と設計
- 持ち帰ってほしいこと

- 前提: Fluentdとその攻撃面
- 何が起きたか: 4件の脆弱性
- 舞台裏: 報告から公開、DB反映まで
- 設定診断ツールを作った判断と設計
- 持ち帰ってほしいこと

Fluentdが実現する世界



もとは、閉じたネットワークで使う前提

これまで

信頼できる送信元

自分たちのサーバから受け取る

いま

信頼できない送信元も

コンテナやクラウドで、境界があいまい

「設定すれば安全」から「既定で安全」へ

Fluentdは外部からデータを受け取る

送信側
タグは送る側が決める

→

Fluentd
in_forward / in_http

→

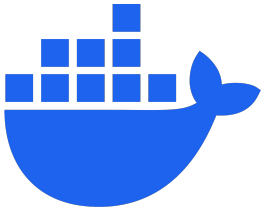
出力
${tag}でパスやURLを組み立てる

信頼できない送信元なら、タグも攻撃者の入力になる

修正の届け先は配布形態の数だけある

- Ruby gem
- Package
  - Fluent Package (fluent-package)
- Container
  - Docker images
  - Daemonset for Kubernetes






LTS版: EOLの系列には修正版が出ない

Fluentd v1.16.x

Fluent Package v5.0.x

2025年末でEOL

修正版は出ない

Fluentd v1.19.x

Fluent Package v6.0.x

2027年末(予定)まで

セキュリティー・バグ修正のみ

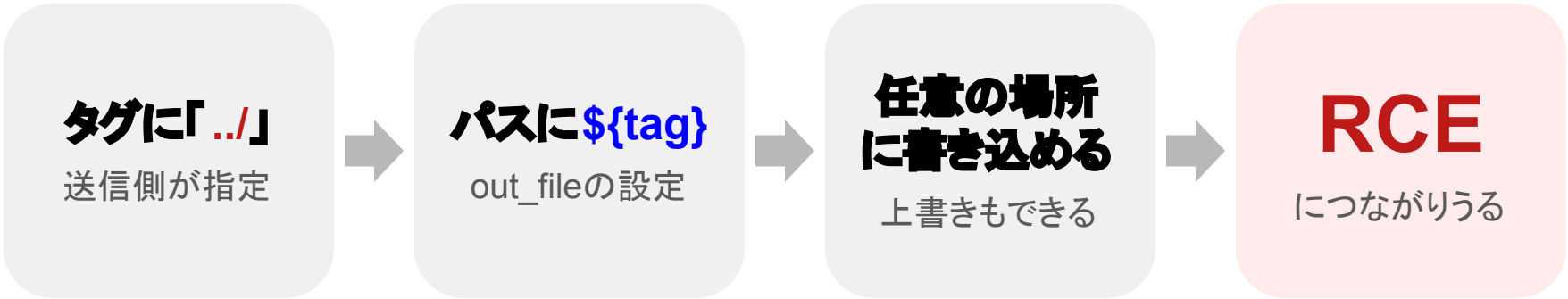
- 前提: Fluentdとその攻撃面
- 何が起きたか: 4件の脆弱性
- 舞台裏: 報告から公開、DB反映まで
- 設定診断ツールを作った判断と設計
- 持ち帰ってほしいこと

Fluentd v1.19.3で修正した4件の脆弱性

CVE	CVSS	内容
CVE-2026-44024	9.8	${tag}を使ったファイルパスへの任意ファイル書き込み（ RCE）
CVE-2026-44025	7.5	monitor_agentのAPIからの機密情報の露出
CVE-2026-44160	7.5	in_http / in_forwardのgzip展開によるメモリ枯渇（DoS）
CVE-2026-44161	7.2	out_httpのプレースホルダ展開による SSRF

※ CVSSは各GitHub Security Advisoryの評価

CVE-2026-44024: タグを悪用してRCEへ



CVSS 9.8 **認証なしで、ネットワークから攻撃できる**

CVE-2026-44025で、弱点が丸見え

設定が見える

CVE-2026-44025:
monitor_agent
から設定情報が漏れる

弱点が分かる

pathに**${tag}**を使う
狙える出力先が見つかる

狙い撃ちできる

CVE-2026-44024:
細工したタグを
送るだけ

設定が丸見えなので、攻撃者は容易に弱点を特定できる

4件とも、影響の有無は設定で決まる

CVE	影響を受ける設定
CVE-2026-44024	${tag}をパスに使い、信頼できない送信元から受信
CVE-2026-44025	monitor_agentのAPIに、外部からアクセスできる
CVE-2026-44160	in_http / in_forwardを、信頼できない送信元に公開
CVE-2026-44161	out_httpのendpointに、${tag}などを使っている

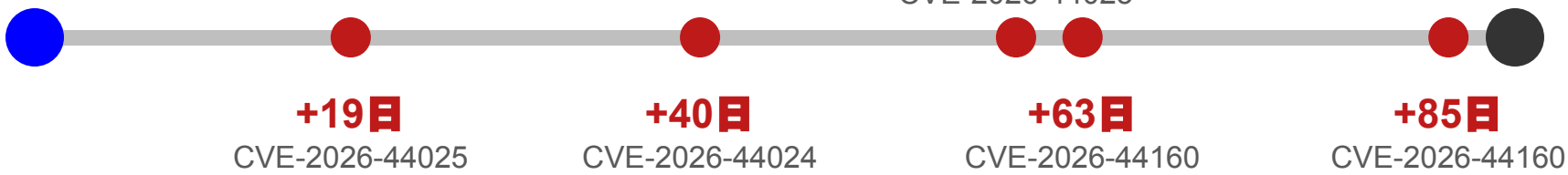
バージョンだけでは、影響を判断できない



- 前提: Fluentdとその攻撃面
- 何が起きたか: 4件の脆弱性
- 舞台裏: 報告から公開、DB反映まで
- 設定診断ツールを作った判断と設計
- 持ち帰ってほしいこと



報告: 同じ脆弱性が複数人から届いた



公開までの 89日間に、ほかの人から 5件の重複報告

最後の重複報告は、公開の 4日前

トリアージ: リリース当日まで続いた

主張を検証

再現手順やPoCをもとに

重複を照合

概要やPoCをもとに

CVSSを決める

CVSS v3の基本評価基準

採番と開示調整: GitHubでCVE IDを取得

5/4〜5　CVE IDを取得（申請から約2週間）

報告者と、公開日とクレジットを調整

6/26　4件のアドバイザリを公開

CVE IDは、GitHub Security Advisoryから取得（CNAはGitHub）

修正: 修正方針と配布形態への展開

ClearCode

Fluentd
v1.19.3



fluent-package（LTS）	**6/26** v6.0.4
Dockerイメージ	**6/26** v1.19.3-2.0
DaemonSet for Kubernetes	**6/30** v1.19.3-1.0

日本の利用者向けに、IPAにも届け出た

ClearCode

GitHub

Security Advisory

Dependabotの利用者に

CVE

cve.org / NVD

各種スキャナに

IPA → JVN

JVN#36011274（6/29）

国内の利用者に日本語で

利用者層に合わせて、届け先を増やす

IPAへの届出は、GitHubのCVE申請とは別の手続き

公開後、CVEはCVE Listに載らなかった

6/26 アドバイザリ公開（GitHub側は即日反映）

7/7 issue報告で、はじめて気づく

CVEレコードは「**存在しない**」

GitHubサポートに問い合わせ

7/8 cve.org / NVDに掲載（公開から12日後）

サポートの回答: 個別には早められない

Fluentdチーム

4件とも、cve.orgでは「存在しない」のままです
GitHubのCVE窓口へのメールも、届きませんでした

GitHub Support

受付順に処理していて、約3週間かかります
個別に早めることはできません

Dependabotのアラートは、アドバイザリをもとに出ます

※ GitHub Supportとのやりとり（2026年7月）を要約

GitHub側とNVD側で、反映日がずれた

	GitHub側	NVD側
データベース	GitHub Advisory Database	cve.org / NVD
反映日	2026年6月26日（公開当日）	2026年7月8日（12日後）
照合できる日	2026年6月26日（公開当日）	**2026年7月13日（17日後）**
参照するツールの例	Dependabot、Trivy Grype、OSV-Scanner	CPEで照合するツール （Dependency-Trackなど）

※ 日付はUTC。照合できる日は、NVDでCPE設定が追加された日（CVE-2026-44024の例）

v1.19.3の修正に、すぐ穴が見つかった

6/26
v1.19.3で公開

9/29
v1.19.4で公開



+3日・+7日
6/29・7/3

+56日
8/21

+79日
9/13

+89日
9/23

6人が、CVE-2026-44024の修正漏れ(「..」タグ)を報告

最初の報告は、v1.19.3公開の当日

ほかにも、CVE-2026-44160の修正漏れが2人、in_syslogのバッファが2人(同じ日に約4時間差)

続き: v1.19.4で、さらに4件を修正

GHSA	CVSS	内容	CVE
GHSA-5hq3-r276-rfr5	7.5	CVE-2026-44024の修正漏れ: 「..」タグでパストラバーサル	未採番
GHSA-h3xv-5jpx-r4j2	7.5	in_syslog（TCP/TLS）の受信バッファでメモリ枯渇	未採番
GHSA-g69w-f42r-xp35	7.5	in_httpのndjson解析でメモリ枯渇	未採番
GHSA-x455-r5cg-h9p9	2.9	CVE-2026-44160の修正漏れ: バッファ展開でメモリ枯渇	未採番

※ 4件とも2026年9月29日にアドバイザリを公開

AIスキャンの結果が、報告と一致した

ClearCode

4件中3件

v1.19.4で直した脆弱性を、Claude Securityも指摘していた
（v1.19.3リリース後 ～ v1.19.4リリース前の期間にスキャンした）

AIで、誰でも同じ穴を見つけられる



v1.19.4: 申請から2か月、CVEが付かない

	v1.19.3（6月）	v1.19.4（9月）
CVE ID	5月4〜5日に予約	9月8日に申請、未採番
アドバイザリ公開	6月26日（CVE付き）	9月29日（CVEなし）
cve.org / NVD	7月8日に掲載	掲載なし

リリースのお知らせも、いつものCVEではなく、アドバイザリへのリンクで案内した

※ 2026年11月17日時点の状況

背景: GitHubへのCVE申請が急増

約4,000件

2026年5月のCVEリクエスト

約10倍

前年比

4月中旬から、処理が当初の目標に追いついていない

出典: GitHub Blog「Inside the Advisory Database and what happens when vulnerability volume breaks records」（2026年6月29日）

Fluentdだけではない: Railsも同じ

21件

11/4のRailsのセキュリティ修正
多くが「No known CVE」

CVEなし

CVSS 9.8の脆弱性も
（Active Storage）

3月のリリースでは、予約から 6日でCVEが公開されていた

リリースノートのリンク先でも、**CVE IDは「なし」**（例: GHSA-f3w6-w445-fwh9）

出典: Rails公式ブログ「Rails Versions 8.0.5.2 and 8.1.4.1 have been released!」（2026年11月4日）、Rescanaの解説、CVE-2026-33195のCVEレコード



CVEのない修正漏れは見逃される

CVEだけを見るスキャナ

v1.19.3は「問題なし」

NVDのCVEは「v1.19.3で修正済み」のまま

実際は

v1.19.3にも修正漏れ

GHSA-5hq3（CVE-2026-44024）
GHSA-x455（CVE-2026-44160）

自社のツールは、

どちらを見ていますか？



- 前提: Fluentdとその攻撃面
- 何が起きたか: 4件の脆弱性
- 舞台裏: 報告から公開、DB反映まで
- 設定診断ツールを作った判断と設計
- 持ち帰ってほしいこと

なぜ作ったか: バージョン判定の限界

バージョンで判定

すべて「影響あり」

v1.19.4未満なら、設定に関係なく

設定で判定

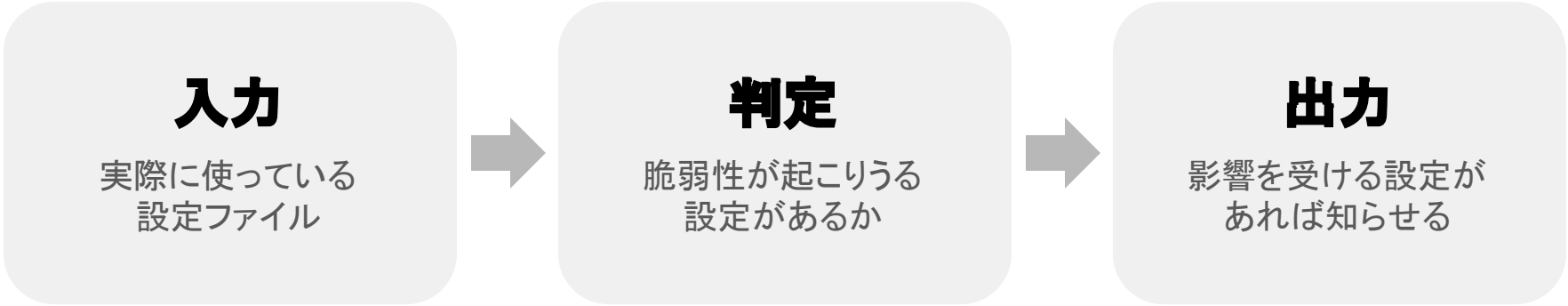
対象を絞り込める

8件とも設定次第。設定で当たりを付ける

判定手段を、プロジェクト側から提供する



設計: 設定と脆弱性の影響条件を照合する



見逃さないことを重視

運用条件によっては、問題ない場合も検知される

https://github.com/clear-code/fluent-vulnerability_checker

fluent-vulnerability_checkerの実行例

```
$ fluent-gem install fluent-vulnerability_checker
$ fluent-vulnerabilitycheck -c /path/to/your/fluentd.conf
```

```
# Exposure of Sensitive Information via Monitor Agent API
* Severity:   High
* CVSS Score: 7.5
* Location:   <source> - @type monitor_agent
* State:      bind = '0.0.0.0'.
              (Note: In Fluentd 1.19.2 and earlier, the default value of bind is 0.0.0.0.)
* Description:
  * JA: monitor_agentによって公開されているAPIを悪用して、Fluentdの設定内容を読み取られる危険性があります。
* Workaround:
  * JA: すぐにパッケージを更新できない場合、アクセスの制限、ローカルホストからのみのアクセスを受け付けるなどの
        緩和策を適用してください。
```

例: monitor_agentを外部に公開している設定（CVE-2026-44025）を検知。英語の説明は省略

- 前提: Fluentdとその攻撃面
- 何が起きたか: 4件の脆弱性
- 舞台裏: 報告から公開、DB反映まで
- 設定診断ツールを作った判断と設計
- 持ち帰ってほしいこと

崩れつつある、これまでの前提

立場	項目	これまで	いま
メンテナー	報告	少数の報告を1件ずつ検証	同じ脆弱性が複数人から同時に届く
	採番	数日でCVEが付く	2週間かかり、今は2か月付かない
	相談先	CNAにメールで相談	メールは届かず、「早められない」
インフラ	検知	新しいCVEをスキャナが拾う	CVEなしの脆弱性は、NVD側に出ない
	修正の確認	修正版に上げれば終わり	修正漏れの公表がないか、継続して確認
	更新の頻度	セキュリティ更新は年数回	数か月おきに続く（Fluentd・Rails）

利用者として: CVE起点の運用を見直す

1 **起点を、GHSA / OSVに移す**
CVEがなくても載るデータベースを見る

2 **GHSA IDでも管理する**
チケットやSLAを、CVE IDだけに紐付けない

3 **上流を直接購読する**
重要なOSSは、アドバイザリやリリース告知を

利用者として: 影響と優先順位を判断する

ClearCode

参照するDBを確認

GitHub側か、NVD側か

配布物の版で確認

本体でなく、パッケージやイメージの版

設定で優先度を付ける

診断ツールで、使っている設定を調べる

EOLの系列は移行する

fluent-package v5など

手順の詳細は、ブログにまとめた

ククログ（2026年10月30日）

GitHub Security Advisoriesを使った OSSの脆弱性対応手順

（Fluentd v1.19.3・v1.19.4の実例）

プライベートフォークでの修正、 CVEの申請、IPAへの届出まで

https://www.clear-code.com/blog/2026/10/30/fluentd-security-advisory-workflow.html

まとめ

起きていること	こうする
影響は設定次第	**診断ツールで設定を調べる**
CVEは遅れる、付かないことも	**GHSA / OSVを起点にする**
AIで、誰でも同じ穴を見つける	**上流のアドバイザリを直接購読**
修正版にも修正漏れがありうる	**修正漏れの公表を継続して確認**

CVEだけを起点にすると、取りこぼす

いまも新たな脆弱性に対応中
リリース時に、今日の観点で確認を

Discussionに日本語Q&Aあります！

https://github.com/fluent/fluentd/discussions/categories/q-a-japanese

日本語での問い合わせも、

お待ちしてます！！

Issue報告やPull Requestもお待ちしてます！

脆弱性の報告はSecurityタブから非公開で

