January 2023

Using federated social networks for efficiently distributing CVE information

Armijn Hemel

Using federated social networks for efficiently distributing CVE information

Abstract

This document describes an open method for distributing information related to CVEs using social networks (in particular the open standard ActivityPub [2]), which would allow users to get updates for specific CVE reports or specific events, and also potentially capture feedback from users around these updates.

The CVE [1] reporting system is the main vehicle for openly distributing information related to security vulnerabilities in software or configurations of software. CVE reports are useful but it has proven to be difficult to easily inform users about updates to CVE reports. In response a whole industry has sprung up around CVE and aggregation and distribution of information related to CVEs to fill this gap. The platforms used for distributing the aggregated information are closed and require payment to access the information.

Keywords

activitypub, fediverse, cve, cwe, security, social networks, activitystream

Background

CVE reports are one of the main mechanisms to distribute information related to security vulnerabilities in software, or configurations of software. The CVE format was created in 1999 to add structure to vulnerability reports which before CVE had not been standardized.

Although very helpful there are drawbacks to the current way that CVE information is distributed and stored:

1. the information is effectively under control of a single government (USA) that can influence if CVE reports are kept under embargo or not
2. CVE information is somewhat silently updated and not (publicly) versioned: if there are updates then the CVE sources are silently updated and if you want to stay up to date then you have to regularly visit the CVE website and check if any edits have been made. There are feeds that list the modifications of the last 7 days, but a user still has to actively seek and process these. For older updates the main data files have to be downloaded and processed.

3. getting updates for just the CVE reports that are of relevance is currently not possible: the only option is to consume all of the CVE data, plus possibly any update feeds.
4. getting notifications for CVE reports that have certain properties is not easy. For example, many CVE reports are for open source packages, but the central entry point into the CVE data set is the number of the CVE, not the name, or name + version, of the open source package. If there would be a new vulnerability for a package (example: BusyBox), then the only way to be notified is to parse every new CVE and every updated CVE.

The model described above is a "pull model", where people have to pro-actively visit the CVE website, retrieve and then process CVE data. This results in a lot of wasted effort and it is unsurprising that a whole industry has sprung up, where companies retrieve, process and enrich the CVE data and then lock it in a data silo, with tightly controlled access.

The opposite of the "pull model" is a "push model", where users subscribe to events. Whenever there is an event change it will be reported to the user without the user having to do anything pro-actively.

Events could be anything related to a CVE report or a property of a CVE report. Example events are:

1. new CVE report is created
2. existing CVE report is updated
3. existing CVE report is deleted/withdrawn
4. certain package is named in CVE report
5. new version of package is available that fixes the security bug described in the CVE report

and so on.

There are various implementations of the "push model". One example that would fit our data model well is a social networking site. Examples of social networking sites are Twitter and Facebook. On these sites users can follow other users, receive updates from other users and possible interact with these updates (such as forwarding, liking, replying, and so on).

CVE reports, software packages, other properties in CVE reports, or a combination of them could all be "users" on a social network. Whenever there is an update a new message is posted on the social network and all of the followers of the "user" will see it and be able to interact with it.

One promising social network technology is W3C ActivityPub. ActivityPub is used by social networks such as Mastodon and other social networks in the so called Fediverse [4]. ActivityPub uses a data format called ActivityStreams [3], which, like ActivityPub, is a W3C standard.

In ActivityPub there are two types of communications:

1. client to server: users (either real users or bot) that communicate with a home server to post message, get messages, react to messages, subscribe to other accounts, and so on
2. server to server: servers take data and commands that clients sent or requested and synchronize between each other.

This proposal only focuses on functionality in the "client to server" part of ActivityPub. CVE reports, properties of CVE reports are all users that are registered on a server and that send updates as messages to the outside world. These messages and actions are then distributed using "server to server" functionality.

Method

To implement this functionality there would be three separate tasks:

1. download and parse CVE reports
2. create new users based on CVE reports and properties associated with CVE reports
3. create update messages and post to the (local) server

Download and parse CVE reports

The CVE website provides multiple feeds for data in JSON and XML formats that can be downloaded and which can then be parsed, after which interesting information can be searched or extracted.

1. download the CVE data from the CVE website in JSON or XML form
2. parse the data with a suitable parser (for JSON or XML), for integrity checks and to make data available for further inspection
3. inspect the different fields in the data of the updated and new reports and store interesting information, such as CVE identifier, references to websites with more information, vulnerability severity, kind of problem (CWE identifiers [5]), CPE identifiers, names of packages, version numbers, description, and so on
4. further inspect the "description" field to extract more information, such as source code paths or other file paths

Create new users

After parsing the all the CVE reports and extracting information from the new and updated CVE reports the information should be inspected to see if new "users" should be created on the server. Some possible reasons (but not limited to these) to create a new user could be:

1. a new CVE number
2. a previously unknown software package (for example from the CPE information or from the description)
3. a new category of bugs (for example from the CWE field)
4. a new vendor

and so on.

There are various ways to create new users:

1. manually: the new CVE reports can be inspected manually or scripts could report new unknown data, after which an administrator manually creates the new users
2. automatically: use a script, plus a list of which properties or combination of properties should result in a new user being created

Create update messages

The update messages that will be sent typically are:

1. a short identifier or text, delimited by some characters, so it is easier for a user or program to filter the interesting messages in a larger stream or to process them
2. a short description of human readable text, intended for human consumption
3. (optional) a URL pointing to a resource with more information, such as a website, pictures, structured data, and so on
4. (optional) tags with extra information intended for filtering, comparable to Twitter hashtags

References

[1] https://cve.mitre.org/ and https://en.wikipedia.org/wiki/Common_Vulnerabilities_and_Exposures

[2] https://www.w3.org/TR/activitypub/

[3] https://www.w3.org/TR/activitystreams-core/

[4] https://en.wikipedia.org/wiki/Fediverse

[5] https://cwe.mitre.org/