cve-search - a free software to collect, search and analyse common vulnerabilities and exposures in software

Freedom #0 in action

CIRCL
Computer Incident
Response Center
Luxembourg

Alexandre Dulaunoy

alexandre.dulaunoy@circl.lu

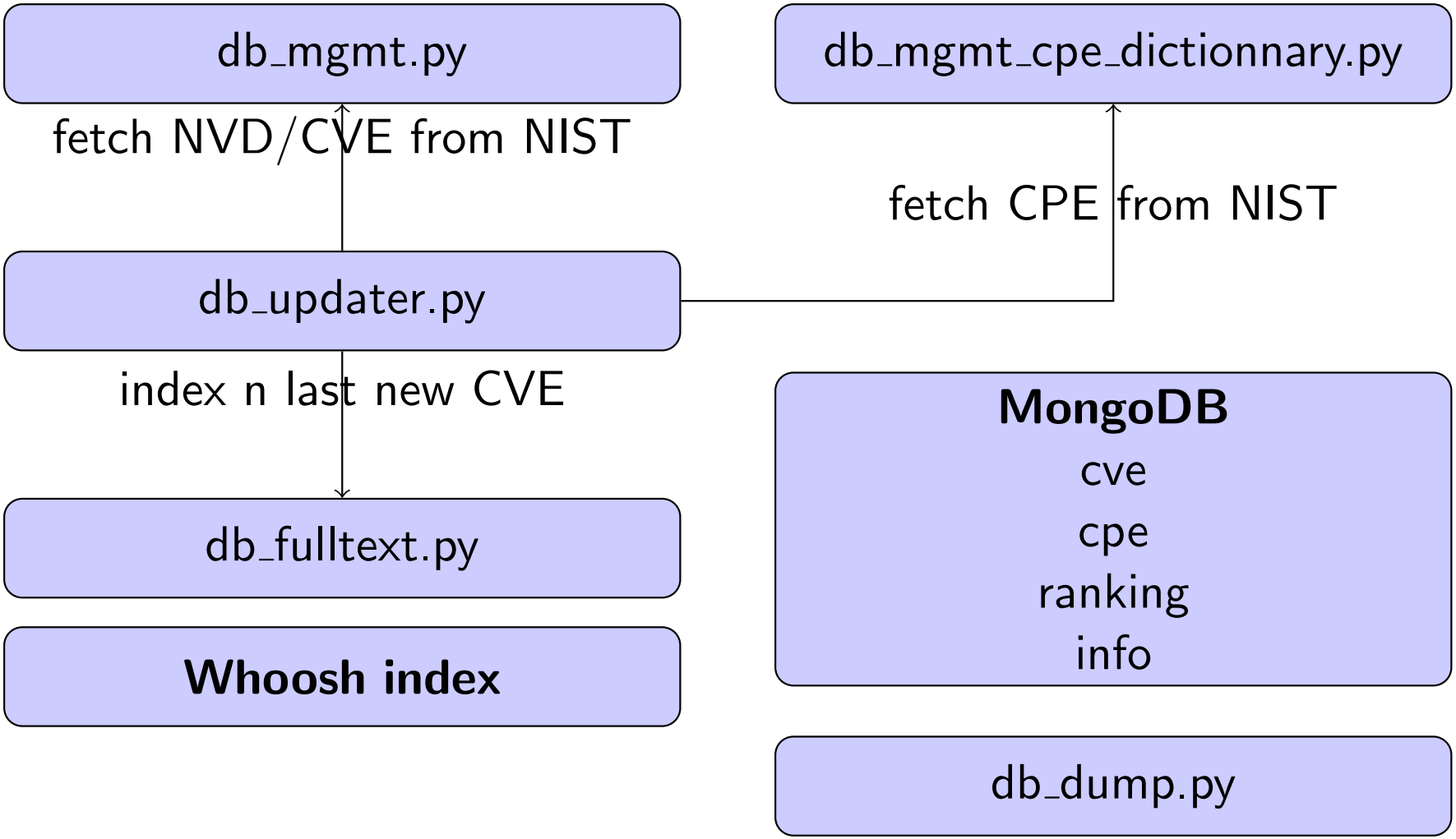
20th December 2013

What we were looking for?

- Local search of common vulnerabilities and exposures ($\rightarrow$ avoid searching NIST/US for your current vulnerable software...).
- Fast-lookup of vulnerabilities (e.g. live evaluation of network traffic for vulnerable software).
- Allow localized classification of vulnerabilities (e.g. classify software following your exposure).
- Flexible data structure (e.g. NIST/NVD is not the only source).
- Allowing the use of Unix-like tools to process the vulnerabilities.

History of cve-search

A functional overview of cve-search (populating database(s)



A functional overview of cve-search (tools)

MongoDB

cve

cpe

ranking

info

search.py

dump_last.py

search_xmpp.py

Whoosh index

search_fulltext.py

cve-search starting up...

Import and update of the CVE/NVD and CPE database:

```
% python3.3 db_updater.py -v -i
```

Search CVE of a specific vendor (via CPE):

```
% python3.3 search.py -p joomla:
...
CVE-2012-5827
CVE-2012-6503
CVE-2012-6514
CVE-2013-1453
CVE-2013-1454
CVE-2013-1455
```

cve-search simple query and JSON output

```
search.py -c CVE-2013-1455 -n
{"Modified": "2013-02-13T13:01:45.353-05:00", "Published": "2013-02-12T20:55:05.387-05:00", "_id": {"$oid": "514cce0db26102134fa3f211"}, "cvss": "5.0", "id": "CVE-2013-1455", "references": ["http://xforce.iss.net/xforce/xfdb/81926", "http://developer.joomla.org/security/news/549-20130202-core-information-disclosure.html"], "summary": "Joomla! 3.0.x through 3.0.2 allows attackers to obtain sensitive information via unspecified vectors related to an \"Undefined variable.\"", "vulnerable_configuration": ["Joomla! 3.0.0", "Joomla! 3.0.1"]}
```

Without CPE name lookup:

```
"vulnerable_configuration": ["cpe:/a:joomla:joomla%21:3.0.0", "cpe:/a:joomla:joomla%21:3.0.1"]}
```

CPE - an overview

```
1 cpe:/{part}:{vendor}:{product}:{version}:{update}:{
      edition}:{language}
```

part	name
o	Operating System
a	Application
h	Hardware

An empty part defines any element. CPE are updated at a regular interval by NIST but it happens that CPE dictionnary are updated afterwards.

Which are the top vendors using the word "unknown"?

```
search_fulltext.py -q unknown -f | jq -r '. | .
    vulnerable_configuration[0]' | cut -f3 -d: | sort |
    uniq -c | sort -nr | head -10
```

Count	CPE vendor name
1145	oracle
367	sun
327	hp
208	google
192	ibm
113	mozilla
102	microsoft
98	adobe
76	apple
68	linux

Which are the top products using the word "unknown"?

```
search_fulltext.py -q unknown -f | jq -r '. | .
    vulnerable_configuration[0]' | cut -f3,4 -d: | sort |
    uniq -c | sort -nr | head -10
```

Count	CPE vendor/product name
191	oracle:database_server
189	google:chrome
115	oracle:e-business_suite
111	sun:jre
101	mozilla:firefox
99	oracle:fusion_middleware
95	oracle:application_server
80	sun:solaris
68	linux:linux_kernel

oracle:java versus sun:jre

```
1 search.py -p oracle:java -o json  | jq -r '.cvss' |
      Rscript -e 'summary(as.numeric(read.table(file("stdin"
      ))[,1]))'
2
3     Min.  1st Qu.   Median     Mean 3rd Qu.     Max.
4     1.80     7.60    10.00     8.45    10.00    10.00
5
6 search.py -p sun:jre -o json  | jq -r '.cvss' | Rscript -
      e 'summary(as.numeric(read.table(file("stdin"))[,1]))'
7
8     Min.  1st Qu.   Median     Mean 3rd Qu.     Max.
9    0.000    5.000    7.500    7.376   10.000   10.000
```

Ranking of vulnerabilities

```
1 db_ranking.py  -c sap: -g accounting -r 3
2 search.py -c CVE-2012-4341 -o json  -r
3 ..."cvss": "10.0", "id": "CVE-2012-4341", "ranking": [[{"
    accounting": 3}]]...
```

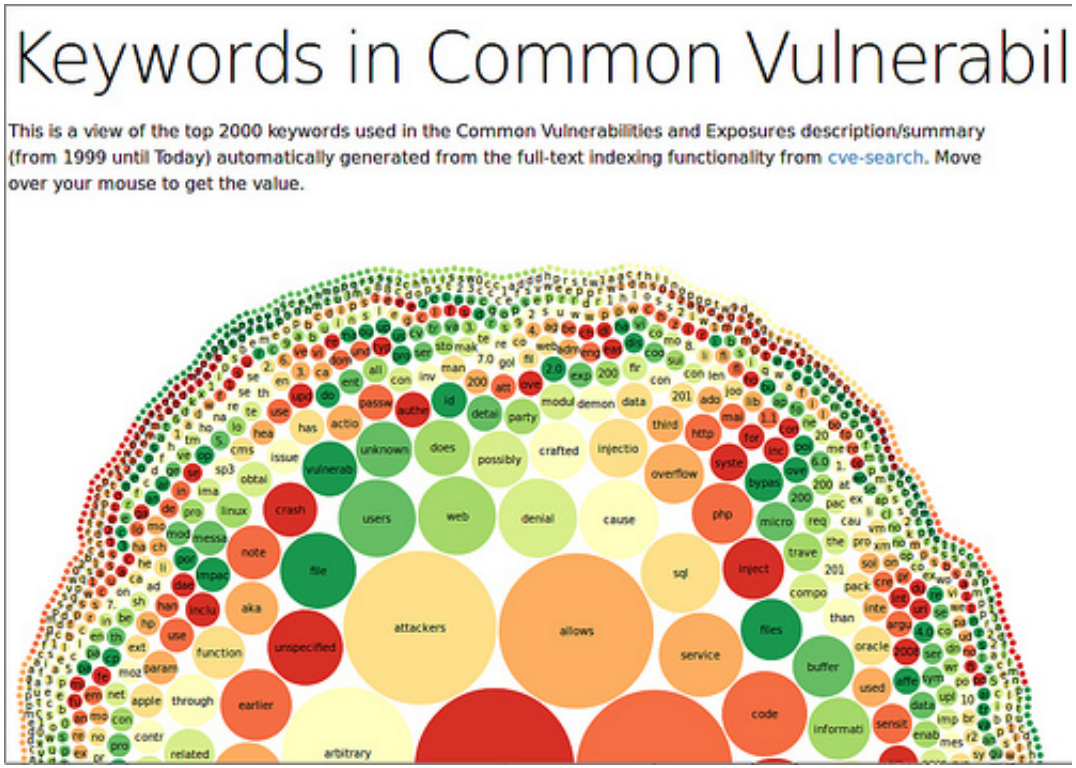
Ranking is simple and flexible. If you are a CSIRT, you can use your own tagging to weight the critical software/vendor in your constituency.

Ranking helping for internal publishing of vulnerabilities

dump_last.py can be used to generate an overview of the current/recent vulnerabilities in your organization. You can limit the result to the ranked software to avoid non-related software vulnerabilities.

```
1 dump_last.py -r -l 100 -f html
2 dump_last.py -r -l 100 -f atom
```

search_fulltext.py -g -s



Freedom #0

FSF Style:

> The freedom to run the program, for any purpose (freedom 0).

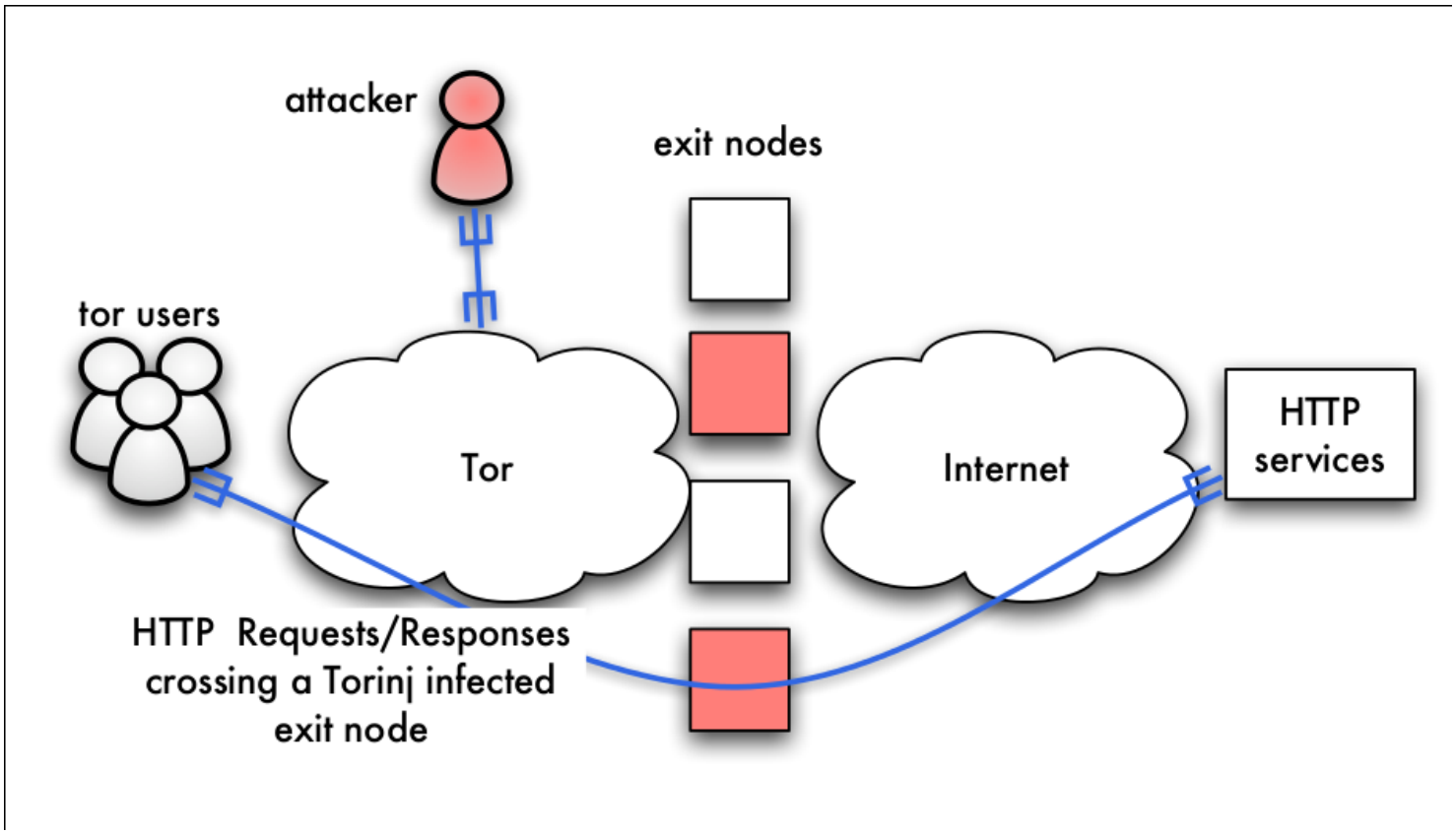
Debian Style:

> No Discrimination Against Fields of Endeavor

Can cve-search be used by bad guys?

- If you know that system is vulnerable, you have two options:
  - If you are a good guy, you inform the system owner to fix the vulnerability.
  - If you are a bad guy, you abuse your position and compromise the vulnerable system.
- cve-search could help both guys.

Where and how to infect random systems? TorInj research project



`http://www.foo.be/torinj/`

TorInj research project and cve-search

- In the initial version of TorInj, the vulnerabilities of the web browsers per client was evaluated offline.
- We were missing a tool to quickly lookup vulnerability per client and find an appropriate automatic exploitation.
- Next version of TorInj is relying on cve-search to do real-time vulnerability lookup, queue and try automatic exploitation.
- The objective of the research is show to the importance of using end-to-end authenticated encrypted communication with Tor and only with up-to-date software.

- Freedom #0 is important. Free software developed for an initial purpose (from defensive to offensive) can be redirected to a new one.
- Looking for open data source of software vulnerabilities to integrate in cve-search.
- Dataset of cve-search can be requested with localized ranking (e.g. per country/region).
- Fork it, abuse it and then send pull request $\rightarrow$ `http://www.github.com/adulau/cve-search`.