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

Alexandre Dulaunoy and Pieter-Jan Moreels

BruCON 0x07

9th October 2015
What we were looking for?

• **Offline** local search of common vulnerabilities and exposures
  ○ → Do you really want to search NIST (based in 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.

• **Build new tools** based on local database of software and hardware vulnerabilities.
History of cve-search

• Wim Remes started with a simple script to read CVE and import it in MongoDB.

• In late 2012, Alexandre Dulaunoy improved the back-end of cve-search and associated tools.

• In 2014, Pieter-Jan Moreels improved the various Web interface to make them usable.

• Today, Alexandre and Pieter-Jan are lead and welcome all additional contributions.
A functional overview of cve-search (populating databases)

- db_mgmt.py
  - fetch NVD/CVE from NIST
- db_updater.py
  - index n last new CVE
- db_fulltext.py
- Whoosh index
- db_mgmt_cpe_dictionary.py
  - fetch CPE from NIST
  - MongoDB
    - cve
    - cpe
    - ranking
    - info
- Redis cache
Data sources imported and used by cve-search

• NIST NVD
  ◦ Common Vulnerabilities and Exposure (CVE), Common Platform
    Enumeration (CPE), Official Vendor Statements, Common Weakness
    Enumeration (CWE), Common Attack Pattern Enumeration and
    Classification (CAPEC), NIST MITRE cross-reference assignment.

• Exploitation reference from D2 Elliot Web Exploitation Framework
  (D2SEC).

• Microsoft Bulletin (Security Vulnerabilities and Bulletin).

• vFeed¹ additional cross-references from Toolswatch.

¹https://github.com/toolswatch/vFeed
A functional overview of cve-search (tools)

**MongoDB**
- cve
- cpe
- ranking
- info

**DB tools**
- db_blacklist.py
- db_cpe_browser.py
- db_fulltext.py
- db_mgmt_.py
- db_notification.py
- db_ranking.py
- db_updater.py
- db_whitelist.py

**Files**
- search.py / search_fulltext.py
- dump_last.py
- search_xmpp.py
- index.py / minimal-web.py
- search_irc.py
- search_cpe.py
- cve_doc.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

```
1 search.py -c CVE-2013-1455 -n
2 {" Modified": "2013-02-13T13:01:45.353-05:00", " Published": "2013-02-12T20:55:05.387-05:00", "_id": {"$oid": "514cbe0db26102134fa3f211"}, "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:

```
1 "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 }}`

<table>
<thead>
<tr>
<th>part</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>o</td>
<td>Operating System</td>
</tr>
<tr>
<td>a</td>
<td>Application</td>
</tr>
<tr>
<td>h</td>
<td>Hardware</td>
</tr>
</tbody>
</table>

An empty part defines any element. CPE are updated at a regular interval by NIST but it happens that CPE dictionary are updated afterwards. `cve-search` supports version 2.2 and 2.3 of the CPE format.
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
```

<table>
<thead>
<tr>
<th>Count</th>
<th>CPE vendor name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1145</td>
<td>oracle</td>
</tr>
<tr>
<td>367</td>
<td>sun</td>
</tr>
<tr>
<td>327</td>
<td>hp</td>
</tr>
<tr>
<td>208</td>
<td>google</td>
</tr>
<tr>
<td>192</td>
<td>ibm</td>
</tr>
<tr>
<td>113</td>
<td>mozilla</td>
</tr>
<tr>
<td>102</td>
<td>microsoft</td>
</tr>
<tr>
<td>98</td>
<td>adobe</td>
</tr>
<tr>
<td>76</td>
<td>apple</td>
</tr>
<tr>
<td>68</td>
<td>linux</td>
</tr>
</tbody>
</table>
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
```

<table>
<thead>
<tr>
<th>Count</th>
<th>CPE vendor/product name</th>
</tr>
</thead>
<tbody>
<tr>
<td>191</td>
<td>oracle:database_server</td>
</tr>
<tr>
<td>189</td>
<td>google:chrome</td>
</tr>
<tr>
<td>115</td>
<td>oracle:e-business_suite</td>
</tr>
<tr>
<td>111</td>
<td>sun:jre</td>
</tr>
<tr>
<td>101</td>
<td>mozilla:firefox</td>
</tr>
<tr>
<td>99</td>
<td>oracle:fusion_middleware</td>
</tr>
<tr>
<td>95</td>
<td>oracle:application_server</td>
</tr>
<tr>
<td>80</td>
<td>sun:solaris</td>
</tr>
<tr>
<td>68</td>
<td>linux:linux_kernel</td>
</tr>
</tbody>
</table>
oracle:java versus sun:jre

```
search.py -p oracle:java -o json | jq -r '.cvss' |
Rscript -e 'summary(as.numeric(read.table(file("stdin"))[,1]))'

    Min. 1st Qu. Median      Mean 3rd Qu.    Max.
  1.80   7.60    10.00     8.45   10.00     10.00

search.py -p sun:jre -o json | jq -r '.cvss' | Rscript -e 'summary(as.numeric(read.table(file("stdin"))[,1]))'

    Min. 1st Qu. Median      Mean 3rd Qu.    Max.
  0.000   5.000    7.500     7.376  10.000     10.000
```
Ranking of vulnerabilities

• Ranking is a simple and flexible approach based on CPE value.
  ○ An organisation or a dept (-g) and an integer value is set when a CPE hits.
• If you are a CSIRT or a local ICT team, you can use your own tagging to weight the critical software/vendor in your constituency.

```python
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 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
Visualization using the browser (index.py)

<table>
<thead>
<tr>
<th>ID</th>
<th>CVSS</th>
<th>Summary</th>
<th>Last (major) update</th>
<th>Published</th>
</tr>
</thead>
<tbody>
<tr>
<td>CVE-2015-0514</td>
<td>5.0</td>
<td>EMC M&amp;R (aka Watch4Net) before 6.5u1 and ViPR SRM before 3.6.1 might allow remote attackers to obtain cleartext data-center discovery credentials by leveraging certain SRM access to conduct a decryption attack.</td>
<td>29-09-2015 - 02:34</td>
<td>21-01-2015 - 16:17</td>
</tr>
<tr>
<td>CVE-2015-0158</td>
<td>4.3</td>
<td>Cross-site scripting (XSS) vulnerability in the Coach NG framework in IBM Business Process Manager (BPM) 8.0 through 8.0.1.3, 8.5.0 through 8.5.0.1, and 8.5.5 through 8.5.5.0 allows remote attackers to inject arbitrary web script or HTML via a crafte.</td>
<td>29-09-2015 - 02:33</td>
<td>24-03-2015 - 03:01</td>
</tr>
<tr>
<td>CVE-2014-9403</td>
<td>4.0</td>
<td>The CWebAdminMod::ChanPage function in modules/webadmin.cpp in ZNC before 1.4 allows remote authenticated users to cause a denial of service (NULL pointer dereference and crash) by adding a channel with the same name as an existing channel but without</td>
<td>29-09-2015 - 02:31</td>
<td>19-12-2014 - 16:59</td>
</tr>
<tr>
<td>CVE-2014-8600</td>
<td>4.3</td>
<td>Multiple cross-site scripting (XSS) vulnerabilities in KDE-Runtime 4.14.3 and earlier, kwebkitpart 1.3.4 and earlier, and kio-extras 5.1.1 and earlier allow remote attackers to inject arbitrary web script or HTML via a crafted URI using the (1) zip.</td>
<td>29-09-2015 - 02:31</td>
<td>08-12-2014 - 12:59</td>
</tr>
<tr>
<td>CVE-2014-7231</td>
<td>2.1</td>
<td>The struts.toml_password function in the OpenStack Oslo utility library, Cinder, Nova, and Trove before 2013.2.4 allows remote attackers to conduct a decryption attack by sending a crafted request using the (1) zip.</td>
<td>29-09-2015 - 02:30</td>
<td>08-10-2014 - 04:15</td>
</tr>
</tbody>
</table>
Optimizing search results - Web interface

github.com/cve-search/cve-search-mt (management tools)
Simple ReST API (minimal-web.py)

API returns JSON data

- Browse vendors (/api/browse).
- Find products associated to a vendor (/api/browse/microsoft).
- Find CVEs for a specific product (/api/search/microsoft/xbox_360).
- Get CVE detailed information including CAPEC and CWE (/api/cve/CVE-2015-0001).
- Recent CVEs (/api/last).

Public version running on https://cve.circl.lu/.

Example:
```
1 curl https://cve.circl.lu/api/last
```
Can cve-search be used by bad guys?

• If you know that a 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\textsuperscript{2}, you abuse your position and compromise the vulnerable system.

• cve-search could help both guys. Don’t forget the freedom 0 of free software \textit{The freedom to run the program, for any purpose.}

\textsuperscript{2}\url{http://www.foo.be/torinj/}
How can you help?

- Looking for open data source of software vulnerabilities to integrate into cve-search.
  - Software or hardware vendors who provide a new open data source are eligible for 1Kg of Belgian chocolate or a pack of 6 Orval beers.
- Dataset of cve-search ranking can be shared with localized information (e.g. per country/region/sector).
- Pushing vendors to release their vulnerability information in an open way.
- Asking vendors to support CPE naming convention (e.g. openssl versus libssl in Debian).
- Fork it, abuse it and then send pull request →
  - github.com/adulau/cve-search (stable)
  - github.com/pidgeyl/cve-search (unstable)
Roadmap and future

• **Add** vulnerabilities data sources from software and hardware vendors.

• **Improve** data structure and back-end to reduce code size.

• **Expand** cve-search to include vulnerabilities without CVE assignment.

• Improve **documentation** and external tools relying on cve-search.
Software using CVE-Search

CVE-Portal
CVE Notification Portal
https://github.com/CIRCL/cve-portal

CVE-Scan
Extract vulnerabilities in systems from NMAP scans
https://github.com/NorthernSec/cve-scan

NorthernSec Vulnerability-Management
Vulnerability management tool

https://github.com/NorthernSec/Vulnerability-management
(Still under development)
Contact Details

Alexandre Dulaunoy

@adulau

a@foo.be

Pieter-Jan Moreels

@PidgeyL

@NorthernSec

pieterjan.moreels@gmail.com