

# vulnerabilità del software

# la fonte del software

- siamo abituati a procurarci il software scaricandolo da Internet
  - es. sistemi operativi, software di produttività, ecc.
- ci fidiamo di chi ci fornisce il software?

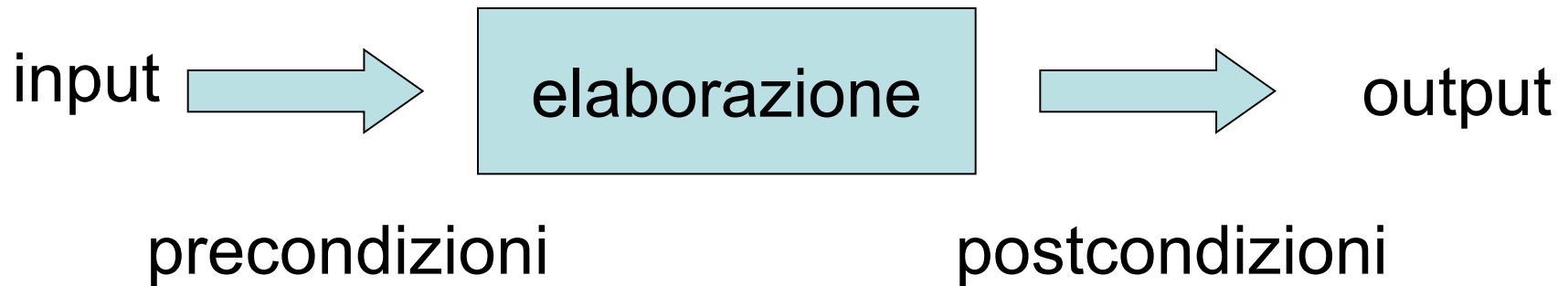
# software da fonte non fidata

- l'esecuzione diretta di software proveniente da una fonte non fidata è una grave minaccia
  - es. software scaricato da siti web malevoli
  - es. “troian” diffusi su social o email
- la vulnerabilità in questo caso non è nei sistemi informatici ma nell'utente inesperto
- contromisure
  - formazione
  - soluzioni tecniche per impedire l'esecuzione del software

# ... ma non basta

- non basta essere certi della fonte
- il software può...
  - contenere errori logici (rispetto ai “requisiti”)
  - non gestire casi limite o input inattesi
  - fare eccessive “assunzioni” sull’ambiente in cui viene eseguito

# correttezza del software



- un programma è corretto quando su qualsiasi input che soddisfa le precondizioni l'output soddisfa le postcondizioni
- assumiamo (leggi “riponiamo fiducia”) che...
  - il produttore/progettista abbia chiare precondizioni e postcondizioni (cioè i requisiti)

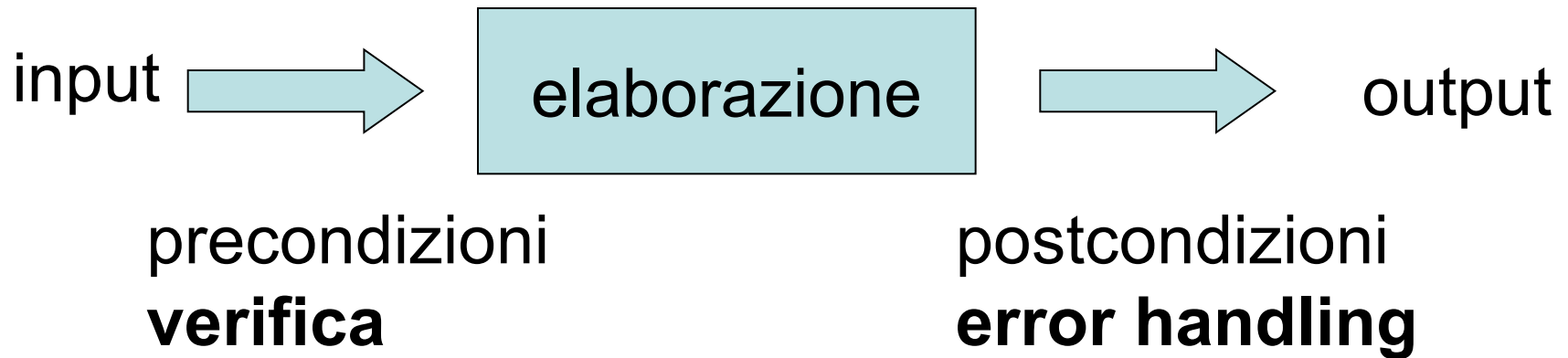
# correttezza e sicurezza

- programmi non corretti sono una minaccia
  - perché fanno cose inattese
- contromisura
  - formazione dei programmatori puntata sulla correttezza rispetto a requisiti descritti formalmente
  - collaudo

# input inattesi

- ...ma la correttezza non basta
- non è detto che l'input soddisfi le precondizioni!

# vulnerabilità: mancata verifica dell'input



- un programma corretto è **vulnerabile** quando esiste un input che **non soddisfa la precondizioni (malformato)** per cui non c'è una verifica e un error handling “adeguato”
  - tipicamente la verifica o non c'è o non rileva tutti gli input malformati

# due approcci opposti «by contract» vs. «defensive»

- contratto tra chiamante e chiamato
  - contratto = preconditione+postcondizione
  - importante nell'ambito della chiamata a metodo (o funzione o affini)
- approccio *design by contract*
  - il chiamato assume che le preconditioni siano rispettate
  - efficiente
  - tipicamente adottato per i rilasci ufficiali
- approccio *defensive programming*
  - il chiamato non si fida e verifica la preconditione
  - inefficiente
  - tipicamente adottato in fase di sviluppo e debug
  - ma anche **fondamentale per la sicurezza**
    - da usare in release solo dove è strettamente necessario (vedi «input non fidato»)

# definizione

## *input fidato e non fidato*

- considera un processo  $P$ 
  - inteso come esecuzione di un programma
- $P$  ha in generale vari input
  - standard input, socket, variabili di ambiente, file, ecc.
- ciascun dato di input ha una sorgente  $S$  (o fonte)
  - cioè un soggetto che ha creato il dato
- $S$  è **non fidata** se  $P$  ha qualche diritto che  $S$  non ha (su almeno un oggetto)
  - $S$  è fidata se  $P$  ha tutti i diritti uguali o minori di  $S$

# dalla vulnerabilità alla minaccia

- un programma vulnerabile diviene una **minaccia** quando il suo input proviene da sorgente non fidata
- in tal caso, la sorgente può sfruttare la vulnerabilità del programma per effettuare operazione che altrimenti non potrebbe fare

# input fidato e non: esempi

- esempi di fonti non fidate
  - pagine web per il browser
    - il browser può scrivere sulla home dell'utente, chi ha creato la pagina web no
  - richieste http per un web server
    - il web server può leggere il filesystem dell'host su cui è installato, il browser che fa la richiesta no
  - email per il mail user agent (mua)
    - il mua può scrivere sulla home dell'utente, chi ha creato l'email no
  - i parametri del comando passwd per il comando passwd
    - il comando passwd può modificare il file /etc/passwd, l'utente che lancia tale comando no (non direttamente)

# possibili effetti di un attacco

- se l'input non è validato il comportamento può essere imprevedibile
- tipicamente crash
  - ...se l'input contiene è errore innocuo
- nel caso peggiore il programma può eseguire operazioni arbitrarie
  - ...per esempio formattare il vostro hard disk
- se l'input è costruito ad arte da un hacker egli può decidere ciò che il programma attaccato eseguirà

# applicazioni comuni e input non fidato

- altri esempi di programmi in cui una vulnerabilità può rappresentare una minaccia...
- ...quando l'input (documenti o programmi) sono ottenuti via email, web, ftp
  - suite di produttività (es. office)
  - viewer (es. acrobat per i pdf)
  - interpreti anche se “sicuri”
    - es. Java Virtual Machine del vostro browser
    - virtualizzazione, sandbox, ecc.

# (interpreti sicuri e sandbox)

- alcuni sistemi eseguono software in maniera da evitare tutti gli effetti collaterali che tale esecuzione può provocare
  - compresi gli effetti di possibili comportamenti malevoli o attacchi
- tale modalità di esecuzione è alle volte detta *sandbox*
- ... ma non è detto che la sandbox non sia esente da bug...

# accertare la presenza di una vulnerabilità dai suoi effetti visibili

- un qualsiasi comportamento anomalo (inatteso) può essere riconducibile ad una vulnerabilità
  - a fronte di un input ben formato o, spesso, malformato

casi notevoli:

- crash
  - tipico di programmi compilati
- errore proveniente dal database
  - tipico delle web application
- errore proveniente dall'interprete
  - per i programmi interpretati

## “...ma è difficile da sfruttare”

spesso ci si chiede se una vulnerabilità sia rilevante in relazione alla difficoltà d'uso da parte di un hacker

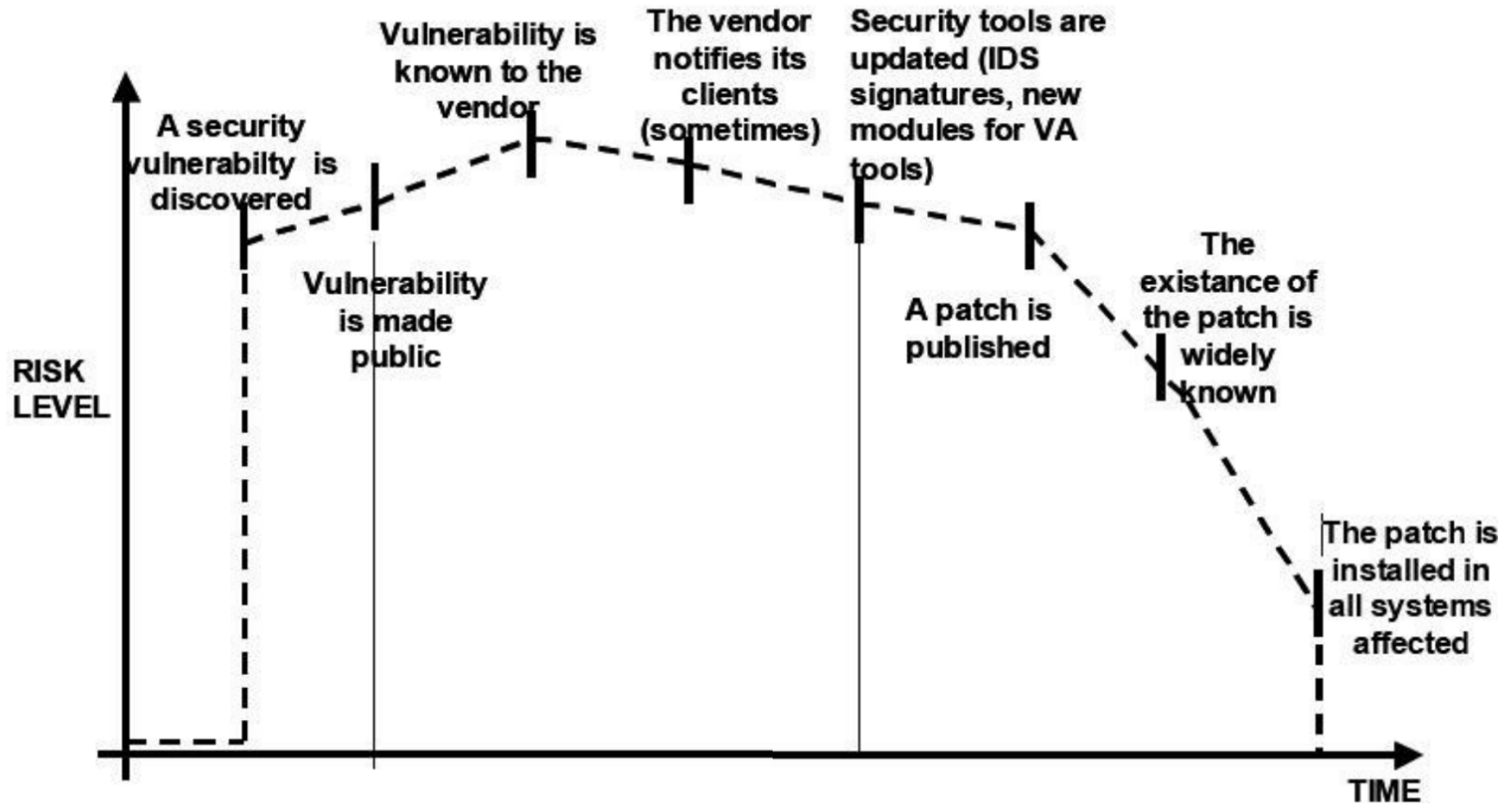
- se è difficile o no da sfruttare non è una questione che compete all'utente
- gli hacker riescono a produrre exploit anche per vulnerabilità apparentemente non usabili

quindi....

# cosa fare in caso di sospetta vulnerabilità

- chi trova una vulnerabilità in un software noto...
  - avvisa il “suo” Computer Emergency Response Team (CERT) o Computer Security Incident Response Team (CSIRT)
- il CERT/CSIRT
  - verifica l’esistenza della vulnerabilità
  - avverte il produttore
  - dopo un certo periodo di tempo (15-30gg) divulga il security advisory (tipicamente via web o mailing list)

# vulnerability lifecycle and “risk” evolution



# il mercato degli zero-day

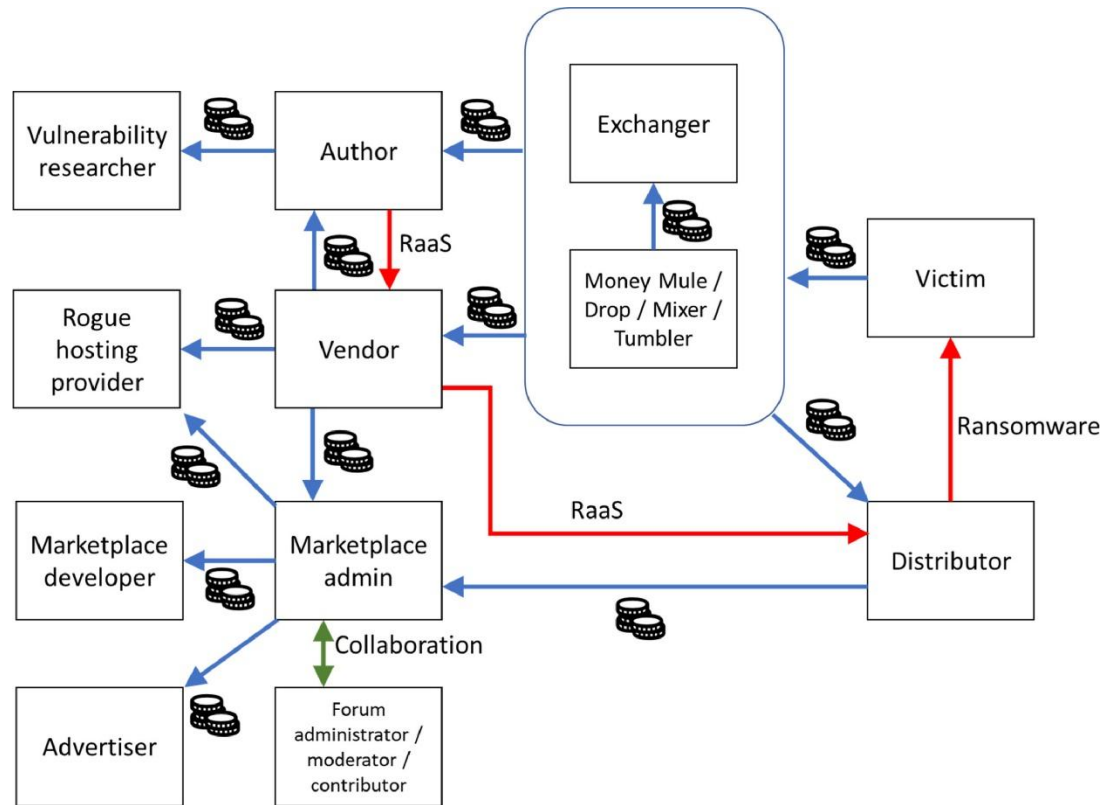
- le vulnerabilità non note sono dette *zero-day*
- uno zero-day è una informazione preziosa per chi sviluppa malware
- gli zero-day possono essere venduti su mercati illeciti nel *dark web* (o *dark net*)
  - il dark web è basato su reti cifrate non tracciabili come ad esempio TOR

# cybercrime market

- anche altri semilavorati o prodotti hanno un mercato illecito
  - exploit
  - virus
  - zombies (macchine compromesse)
  - botnet (insieme di zombies comandate in maniera coordinata), sempre più spesso venduti come \*-as-a-service
  - credenziali per login a vari servizi
  - n. di carte di credito
  - dati personali

# cybercrime value chain

- per esempio



Tratto da  
P. H. Meland, et al., The Ransomware-as-a-Service economy within the darknet,  
Computers & Security, Vol. 92, 2020

# un listino prezzi

Fonte: kaspersky (2009)

- botnet: \$50 to thousands of dollars for a continuous 24-hour attack.
- Stolen bank account details vary from \$1 to \$1,500 depending on the level of detail and account balance.
- Personal data capable of allowing the criminals to open accounts in stolen names costs \$5 to \$8 for US citizens; two or three times that for EU citizens.
- A list of one million email addresses costs between \$20 and \$100; spammers charge \$150 to \$200 extra for doing the mailshot.
- Targeted spam mailshots can cost from \$70 for a few thousand names to \$1,000 of tens of millions of names.
- User accounts for paid online services and games stores such as Steam go for \$7 to \$15 per account.
- Phishers pay \$1,000 to \$2,000 a month for access to fast flux botnets
- Spam to optimise a search engine ranking is about \$300 per month.
- Adware and malware installation ranges from 30 cents to \$1.50 for each program installed. But rates for infecting a computer can vary widely, from \$3 in China to \$120 in the US, per computer.

Una lista più aggiornata: <https://www.privacyaffairs.com/dark-web-price-index-2022/>

# CERT/CSIRT

- i CERT/CSIRT svolgono anche funzioni di coordinamento, divulgazione e supporto alla risposta agli incidenti
  - dovrebbero collaborare tra di loro ma raramente ciò avviene
- cert italiano: [www.csirt.gov.it](http://www.csirt.gov.it)
  - presso l'Agenzia per la Cybersicurezza Nazionale
- lista di CERT famosi  
[https://en.wikipedia.org/wiki/Computer\\_emergency\\_response\\_team](https://en.wikipedia.org/wiki/Computer_emergency_response_team)

# vulnerabilities database

- alcuni database di vulnerabilità famosi
  - National Vulnerability Database - [nvd.nist.gov](http://nvd.nist.gov)
  - Common Vulnerability Exposure - [cve.mitre.org](http://cve.mitre.org)
- altre fonti
  - SANS [www.sans.org](http://www.sans.org)
  - SecurityFocus [bugtraq.securityfocus.com](http://bugtraq.securityfocus.com)
  - tutti i produttori hanno servizi per la sicurezza (mailing list, patches, bugtracking)
    - <http://www.microsoft.com/security>
    - <http://www.redhat.com/security/>

# esempio di security advisory

<https://nvd.nist.gov/search> - search for “explorer jpeg”

## Vulnerability Summary CVE-2005-2308

**Original release date:** 7/19/2005

**Last revised:** 10/20/2005

**Source:** US-CERT/NIST

## Overview

The JPEG decoder in Microsoft Internet Explorer allows remote attackers to cause a denial of service (CPU consumption or crash) and possibly execute arbitrary code via certain crafted JPEG images, as demonstrated using (1) mov\_fencepost.jpg, (2) cmp\_fencepost.jpg, (3) oom\_dos.jpg, or (4) random.jpg.

## Impact

**CVSS Severity:** [8.0](#) (High) Approximated

**Range:** Remotely exploitable

**Impact Type:** Provides user account access , Allows disruption of service

## References to Advisories, Solutions, and Tools

**External Source:** BID ([disclaimer](#))

**Name:** 14286

**Hyperlink:** <http://www.securityfocus.com/bid/14286>

[...]

## Vulnerable software and versions

Microsoft, Internet Explorer, 6.0 SP2

## Technical Details

CVSS Base Score Vector: ([AV:R/AC:L/Au:NR/C:P/I:P/A:C/B:N](#)) Approximated ([legend](#))

Vulnerability Type: Buffer Overflow , Design Error

## CVE Standard Vulnerability Entry:

<http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2005-2308>

# impatto della mancata validazione dell'input e altre vulnerabilità

|                                          | 2012 |        | 2015 |        | 2017  |        | 2019  |        | 2020  |        | 2021 (partial) |        |
|------------------------------------------|------|--------|------|--------|-------|--------|-------|--------|-------|--------|----------------|--------|
| Vulnerabilities for missing input checks |      |        |      |        |       |        |       |        |       |        |                |        |
| XSS                                      | 859  | 14,72% | 992  | 11,63% | 1085  | 6,57%  | 1418  | 8,66%  | 1659  | 9,07%  | 861            | 7,56%  |
| CSRF                                     | 172  | 2,95%  | 285  | 3,34%  | 248   | 1,50%  | 336   | 2,05%  | 353   | 1,93%  | 174            | 1,53%  |
| php                                      | 724  | 12,41% | 646  | 7,58%  | 969   | 5,87%  | 1047  | 6,40%  | 1334  | 7,30%  | 533            | 4,68%  |
| buffer                                   | 436  | 7,47%  | 525  | 6,16%  | 1328  | 8,04%  | 1020  | 6,23%  | 1037  | 5,67%  | 777            | 6,82%  |
| sql                                      | 344  | 5,90%  | 420  | 4,93%  | 671   | 4,06%  | 639   | 3,90%  | 778   | 4,26%  | 454            | 3,99%  |
|                                          | 2535 | 43,45% | 2868 | 33,64% | 4301  | 26,05% | 4460  | 27,24% | 5161  | 28,23% | 2799           | 24,58% |
| Other vulnerabilities                    |      |        |      |        |       |        |       |        |       |        |                |        |
| configuration                            | 129  | 2,21%  | 155  | 1,82%  | 336   | 2,04%  | 590   | 3,60%  | 594   | 3,25%  | 406            | 3,57%  |
| default                                  | 98   | 1,68%  | 104  | 1,22%  | 269   | 1,63%  | 360   | 2,20%  | 404   | 2,21%  | 265            | 2,33%  |
| password                                 | 196  | 3,36%  | 252  | 2,96%  | 432   | 2,62%  | 560   | 3,42%  | 695   | 3,80%  | 297            | 2,61%  |
| firmware                                 | 52   | 0,89%  | 167  | 1,96%  | 447   | 2,71%  | 477   | 2,91%  | 410   | 2,24%  | 239            | 2,10%  |
| android                                  | 168  | 2,88%  | 412  | 4,83%  | 1200  | 7,27%  | 771   | 4,71%  | 695   | 3,80%  | 365            | 3,21%  |
| Total records                            | 5834 |        | 8526 |        | 16509 |        | 16370 |        | 18282 |        | 11385          |        |

da <http://nvd.nist.gov> feeds in formato json

stima in base alla presenza di parole nel campo “description”

alcune righe possono contenere più parole

Le percentuali sono calcolate rispetto al totale dei records