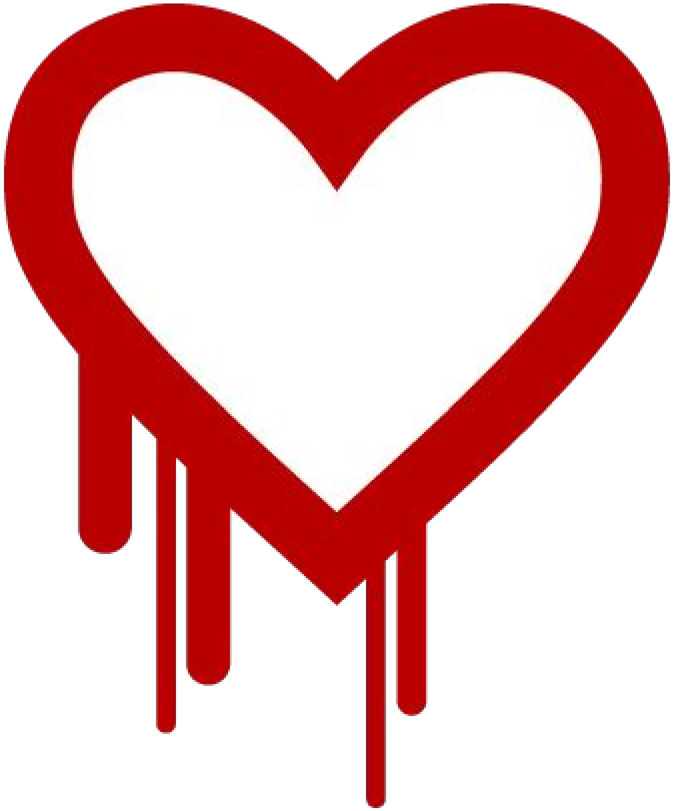
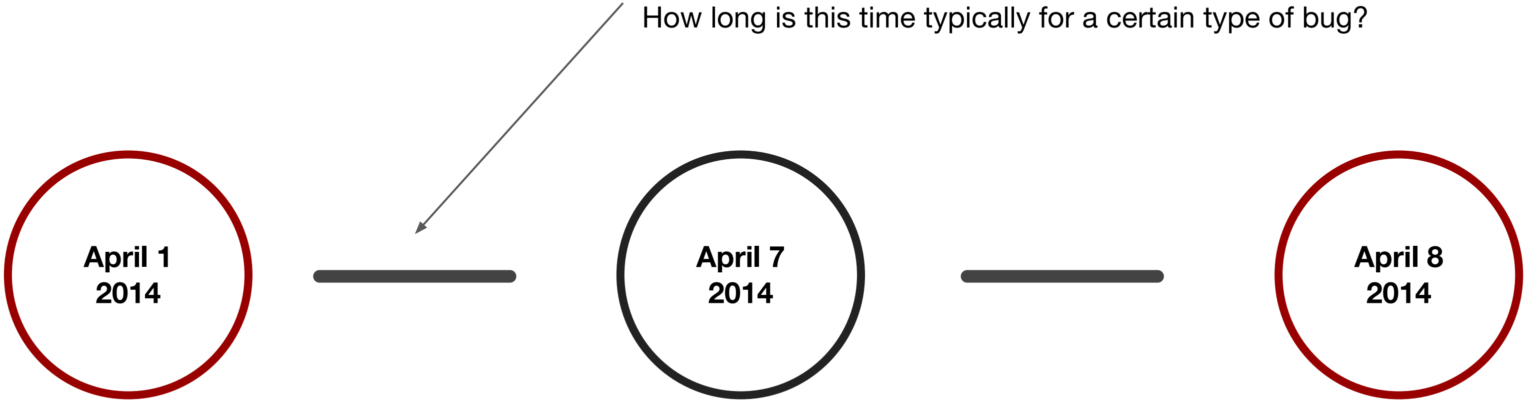


Sources:
https://heartbleed.com/heartbleed.png

How do we know when a fix has been published?
How long is this time typically for a certain type of bug?



April 1 2014

Heartbleed is discovered at Google, the bug was introduced in 2012

~500 000 websites are open to attack

April 7 2014

Fixed openssl library is released

April 8 2014

The Canada Revenue Agency reports a theft of Social Insurance Numbers belonging to 900 taxpayers

Sources:
https://en.wikipedia.org/wiki/Heartbleed#Bugfix_and_deployment

Effects of high-profile incidents on code

Felix Wolff
MA seminar Code Repository Mining
3. semester

Presentation outline

1. Initial research question
    a. Motivation by goto fail; bug
    b. Most high-profile bugs are old and human
2. Findings until now
    a. CVEs - how to name vulnerabilities
    b. Direct reactions to the goto fail; bug
    c. Number of mentions in commit messages per vulnerability name
    d. Commits referring to vulnerability names
3. Alternative research topics
4. Data, structure and sources
5. Next steps

Initial research question

```
static OSStatus
SSLVerifySignedServerKeyExchange(SSLContext *ctx, bool isRsa, SSLBuffer signedParams,
                uint8_t *signature, UInt16 signatureLen)
{
    OSStatus        err;
    …

    if ((err = SSLHashSHA1.update(&hashCtx, &serverRandom)) != 0)
        goto fail;
    if ((err = SSLHashSHA1.update(&hashCtx, &signedParams)) != 0)
        goto fail;
        goto fail;
    if ((err = SSLHashSHA1.final(&hashCtx, &hashOut)) != 0)
        goto fail;
    …

fail:
    SSLFreeBuffer(&signedHashes);
    SSLFreeBuffer(&hashCtx);
    return err;
                                    }
```

Sources:
Apple Open Source Code

Initial research question

People will learn from this and always use braces, right?

```
static
SSLV                    change(SSLContext *ctx, bool isRsa, SSLBuffer signedParams,
                        _t *signature, UInt16 signatureLen)
{




                        .update(&hashCtx, &serverRandom)) != 0)

    if ((er          nSHA1.update(&hashCtx, &signedParams)) != 0)
        goto  ail;
        goto fail;
    if ((err = SSLHashSHA1.final(&hashCtx, &hashOut)) != 0)
        goto fail;
    ...

fail:
    SSLFreeBuffer(&signedHashes);
    SSLFreeBuffer(&hashCtx);
    return err;
                                        }
```

Sources:
Apple Open Source Code

Findings: Most high-profile bugs are old and human

1996 - a rocket of type Ariane 5 crashes when its onboard computer tries to convert a 64 bit floating point number into a 16 bit signed integer. The backup computer crashes at this point, too.

1978 - The F-16 autopilot turns the plane onto its back when crossing the equator. No „negative“ coordinates were considered. This bug was only discovered when the system was tested in simulators.

2009 - Google’s search engine labels every website as malicious, including its own

A good read: https://en.wikipedia.org/wiki/List_of_software_bugs

Findings: Direct reactions to the goto fail; bug

Makefile.in: add -Wunreachable-code

I was reading about the **CVE-2014-1266** SSL/TLS Apple bug on ImperialViolet and learnt that clang has a **separate flag for unreachable code, -Wunreachable-code, that is not included in the -Wall warnings [1]**.

So, let's add -Wunreachable-code to Makefile.in.
[1] https://www.imperialviolet.org/2014/02/22/applebug.htm

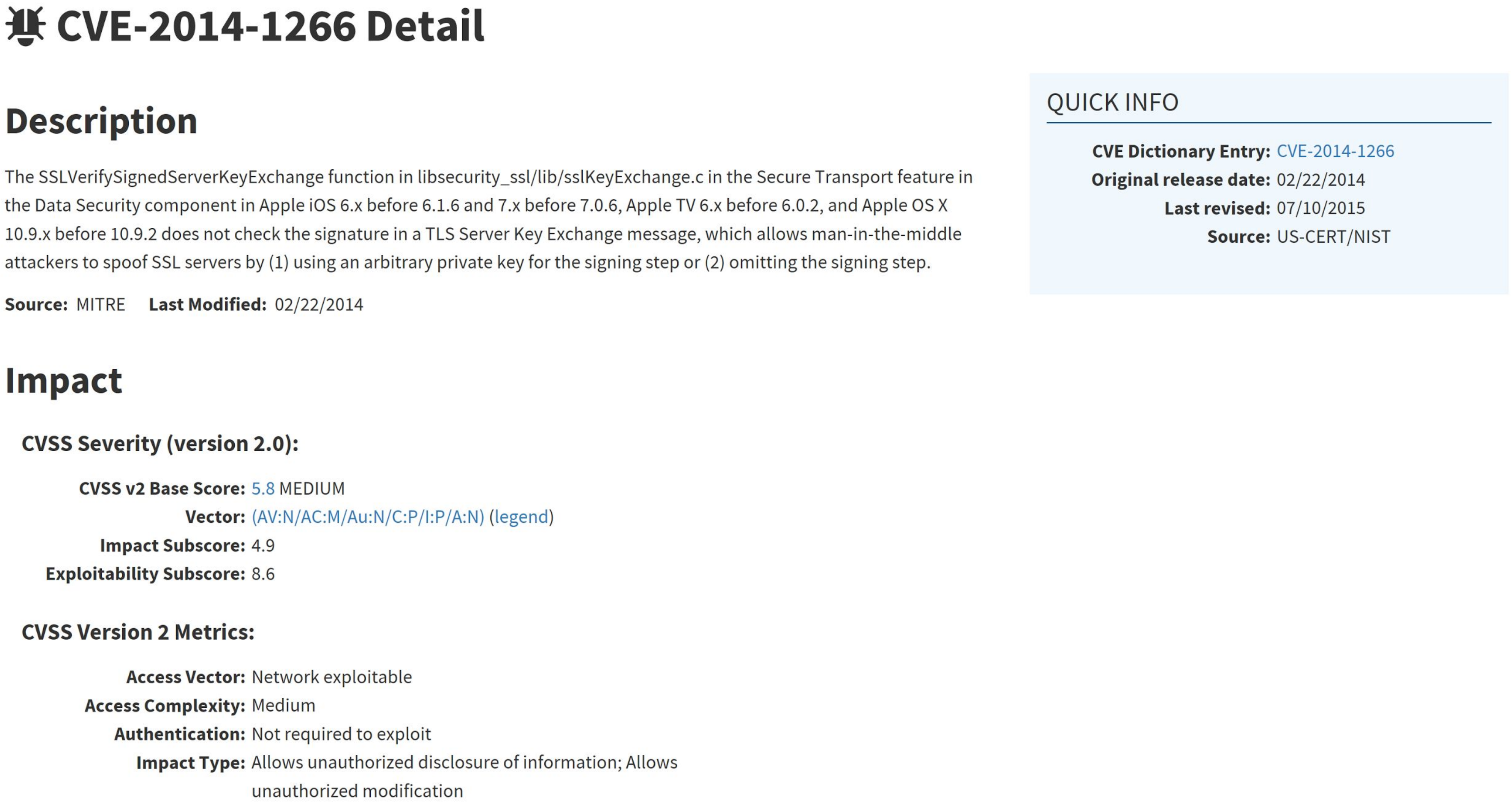
...2000 lines...

-Wmisleading-indentation warns about places where the indentation of the code gives a misleading idea of the block structure of the code to a human reader. For example, given **CVE-2014-1266**:

...2000 lines...

At this point we decided: We ought to look for another approach

CVEs - how to name vulnerabilities



CVE-2014-1266 Detail

Description

The SSLVerifySignedServerKeyExchange function in libsecurity_ssl/lib/sslKeyExchange.c in the Secure Transport feature in the Data Security component in Apple iOS 6.x before 6.1.6 and 7.x before 7.0.6, Apple TV 6.x before 6.0.2, and Apple OS X 10.9.x before 10.9.2 does not check the signature in a TLS Server Key Exchange message, which allows man-in-the-middle attackers to spoof SSL servers by (1) using an arbitrary private key for the signing step or (2) omitting the signing step.

Source: MITRE **Last Modified:** 02/22/2014

QUICK INFO

CVE Dictionary Entry:	CVE-2014-1266
Original release date:	02/22/2014
Last revised:	07/10/2015
Source:	US-CERT/NIST

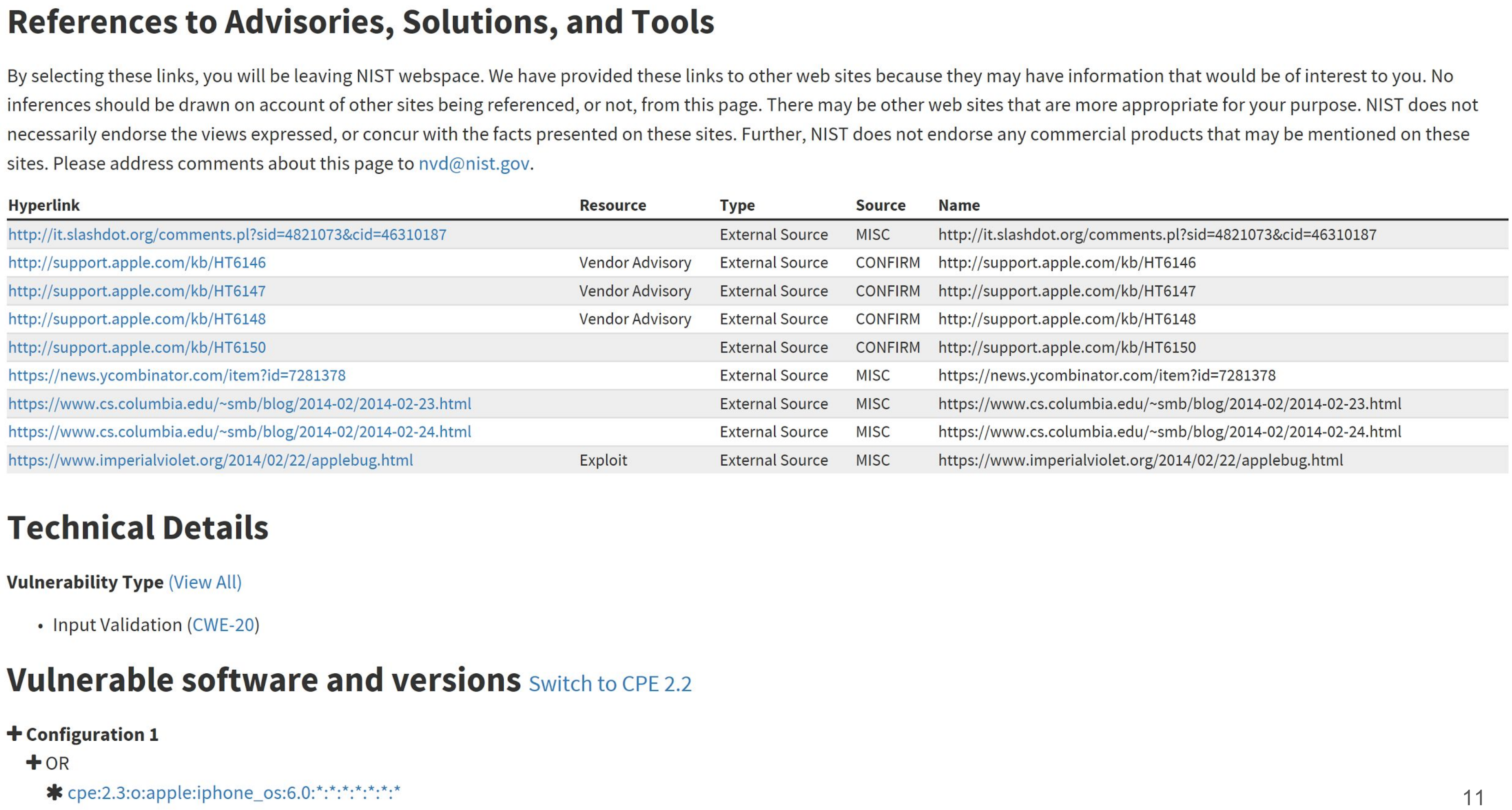
Impact

CVSS Severity (version 2.0):

CVSS v2 Base Score:	5.8 MEDIUM
Vector:	(AV:N/AC:M/Au:N/C:P/I:P/A:N) (legend)
Impact Subscore:	4.9
Exploitability Subscore:	8.6

CVSS Version 2 Metrics:

Access Vector:	Network exploitable
Access Complexity:	Medium
Authentication:	Not required to exploit
Impact Type:	Allows unauthorized disclosure of information; Allows unauthorized modification



References to Advisories, Solutions, and Tools

By selecting these links, you will be leaving NIST webspace. We have provided these links to other web sites because they may have information that would be of interest to you. No inferences should be drawn on account of other sites being referenced, or not, from this page. There may be other web sites that are more appropriate for your purpose. NIST does not necessarily endorse the views expressed, or concur with the facts presented on these sites. Further, NIST does not endorse any commercial products that may be mentioned on these sites. Please address comments about this page to nvd@nist.gov.

Hyperlink	Resource	Type	Source	Name
http://it.slashdot.org/comments.pl?sid=4821073&cid=46310187		External Source	MISC	http://it.slashdot.org/comments.pl?sid=4821073&cid=46310187
http://support.apple.com/kb/HT6146	Vendor Advisory	External Source	CONFIRM	http://support.apple.com/kb/HT6146
http://support.apple.com/kb/HT6147	Vendor Advisory	External Source	CONFIRM	http://support.apple.com/kb/HT6147
http://support.apple.com/kb/HT6148	Vendor Advisory	External Source	CONFIRM	http://support.apple.com/kb/HT6148
http://support.apple.com/kb/HT6150		External Source	CONFIRM	http://support.apple.com/kb/HT6150
https://news.ycombinator.com/item?id=7281378		External Source	MISC	https://news.ycombinator.com/item?id=7281378
https://www.cs.columbia.edu/~smb/blog/2014-02/2014-02-23.html		External Source	MISC	https://www.cs.columbia.edu/~smb/blog/2014-02/2014-02-23.html
https://www.cs.columbia.edu/~smb/blog/2014-02/2014-02-24.html		External Source	MISC	https://www.cs.columbia.edu/~smb/blog/2014-02/2014-02-24.html
https://www.imperialviolet.org/2014/02/22/applebug.html	Exploit	External Source	MISC	https://www.imperialviolet.org/2014/02/22/applebug.html

Technical Details

Vulnerability Type (View All)

- Input Validation (CWE-20)

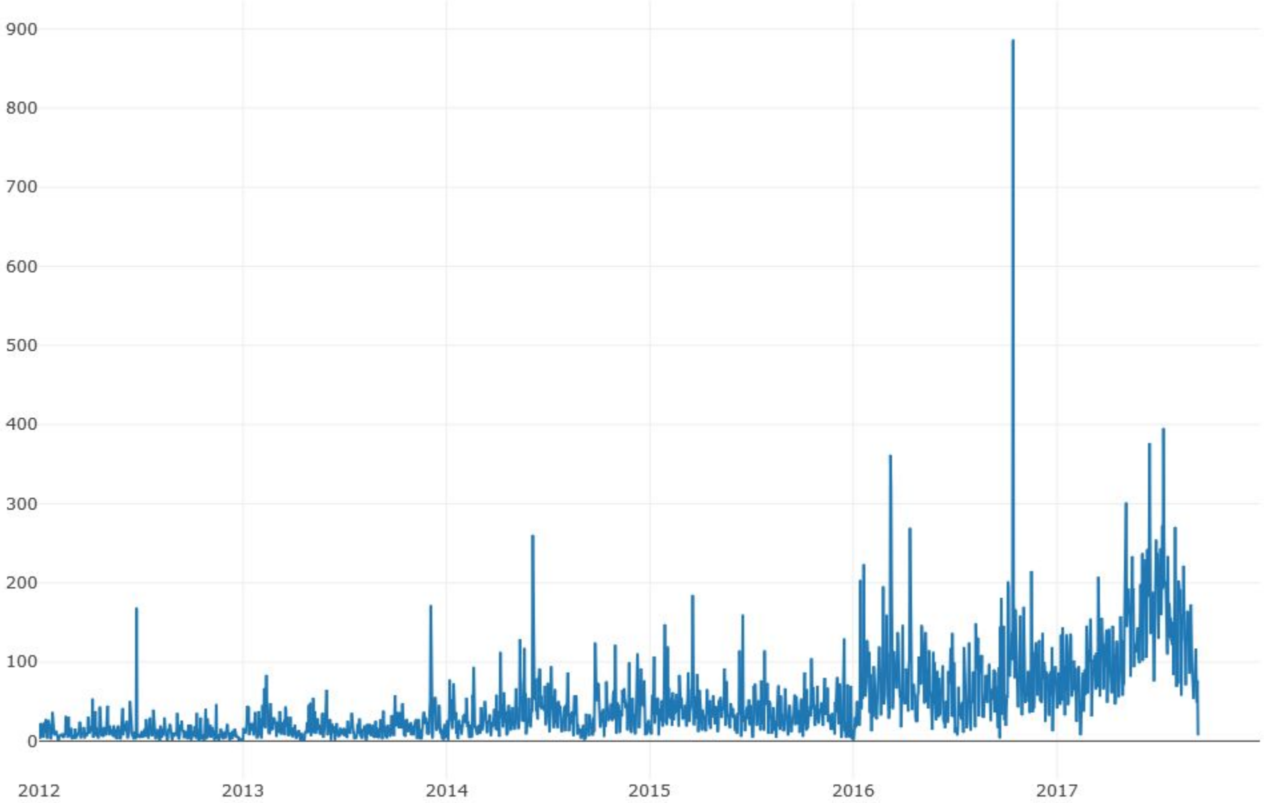
Vulnerable software and versions Switch to CPE 2.2

✚ Configuration 1

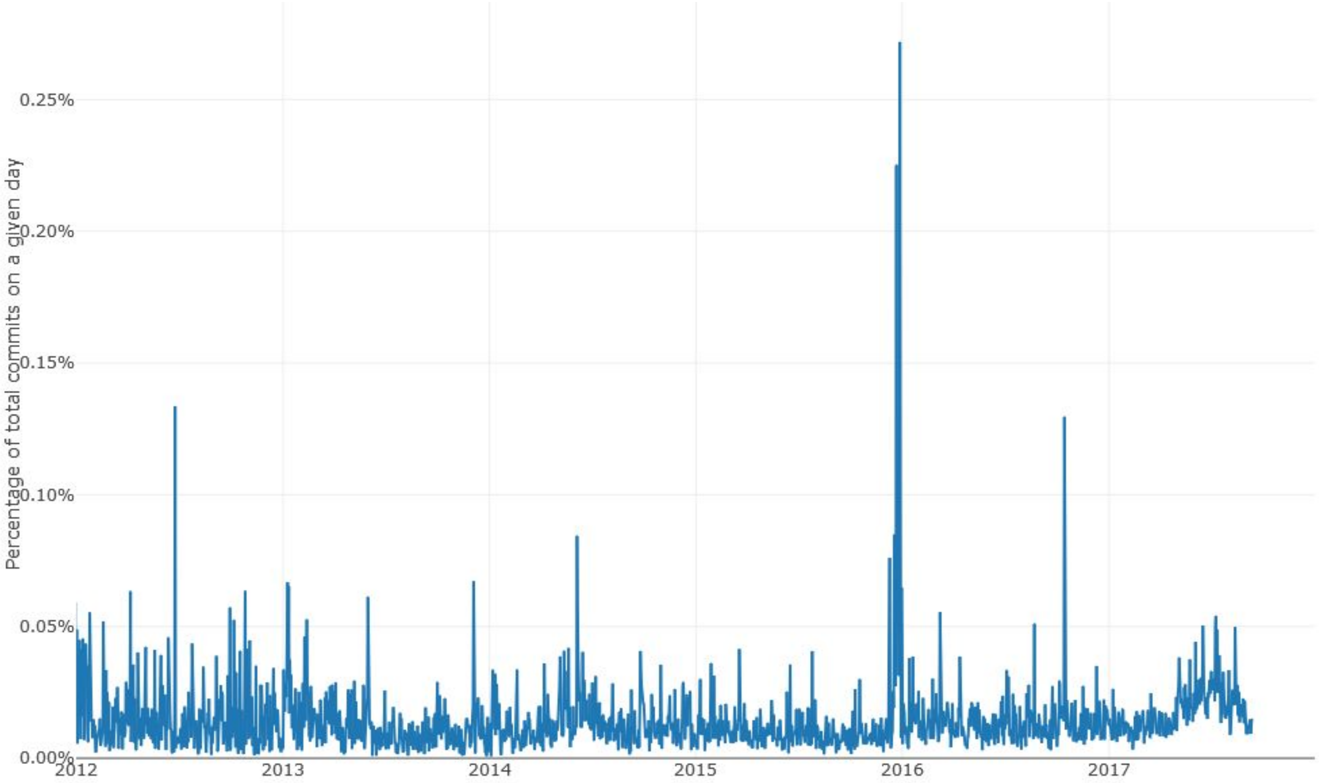
✚ OR

✱ cpe:2.3:o:apple:iphone_os:6.0:*:*:*:*:*:*:*

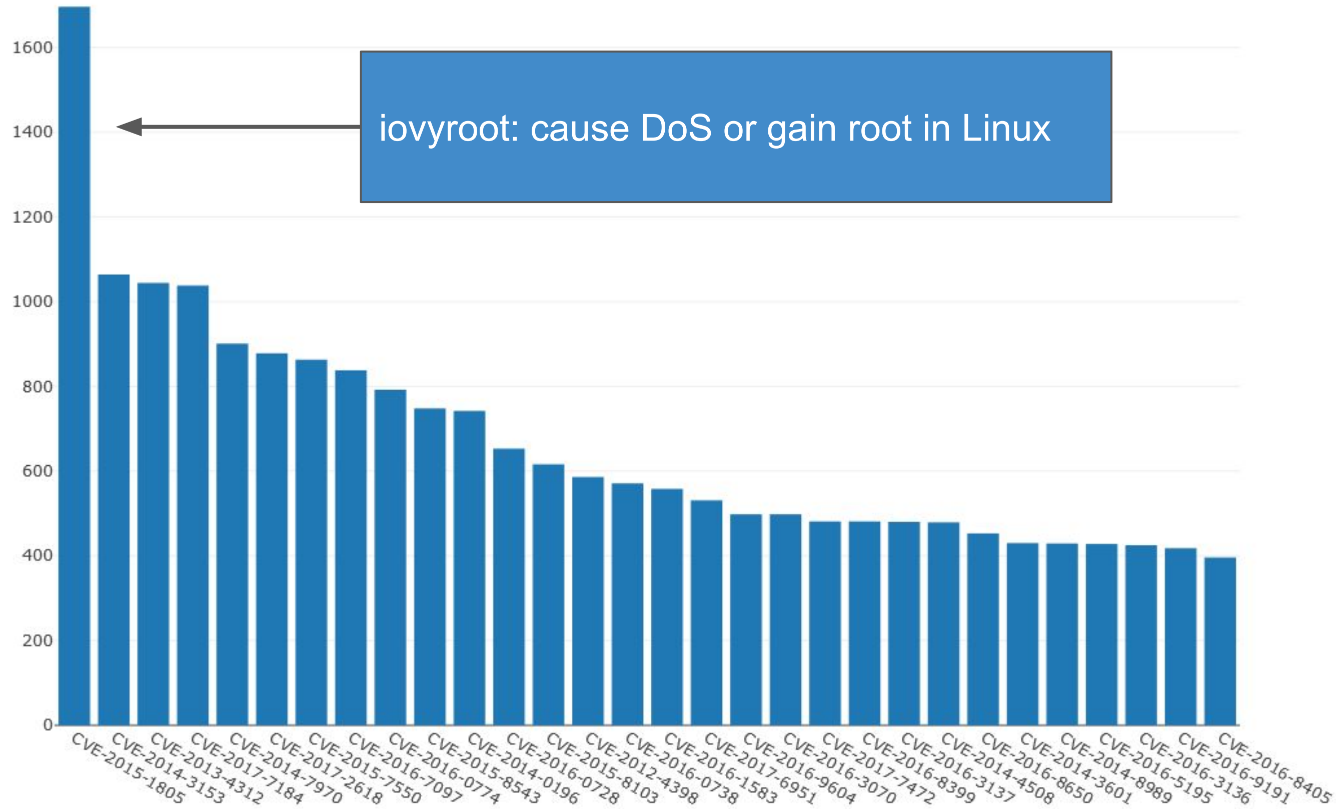
Findings: Commits referring to CVE codes



Commits referring to CVE codes; adjusted



Findings: Number of mentions in commit messages per vulnerability name



iovyroot: cause DoS or gain root in Linux

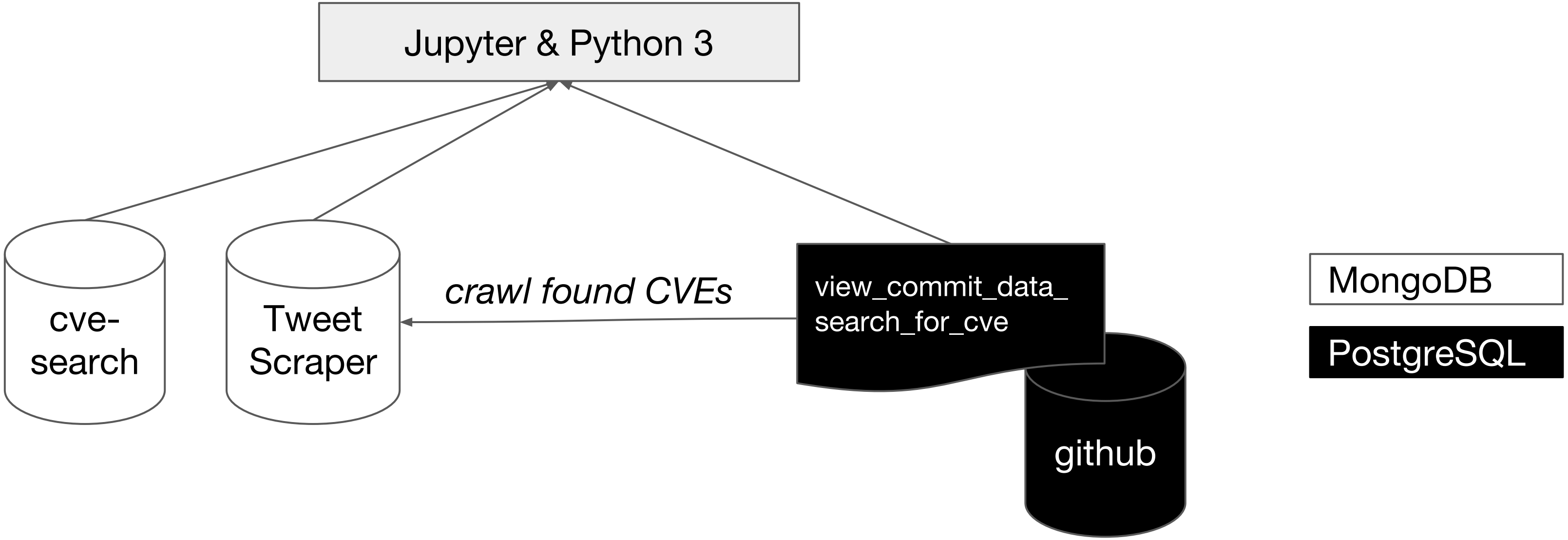
Alternative research questions

There are not enough bugs caused by pure code-related matters in the news, so we came up with new research questions:

1. **Does attention on relevant news channels lead to more reactions in commit messages? If so, why and how? Which channels discuss which type of CVE?**
2. Do bugs and their CVE codes motivate Github Issues, and if so, why?
3. Can reactions to CVEs be categorised? Referring to the previous slide, why are some popular?

Data, structure and sources

Does attention on relevant news channels lead to more reactions in commit messages? If so, why and how? Which channels discuss which type of CVE?



Sources:
https://github.com/cve-search/cve-search
https://github.com/jonbakerfish/TweetScraper

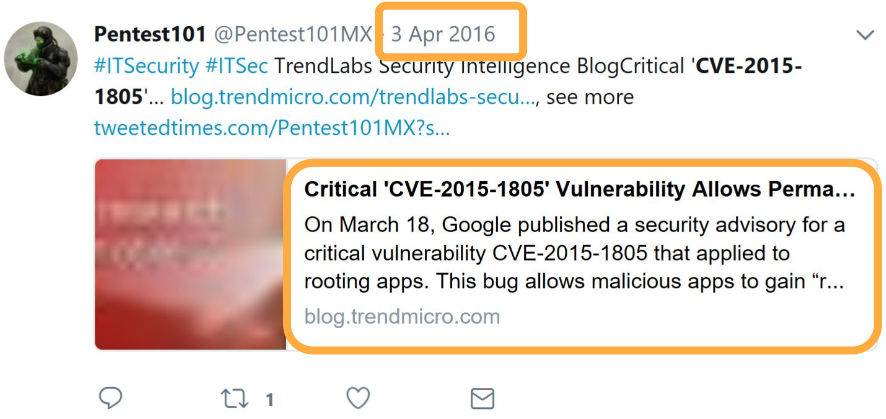
Data, structure and sources

Does attention on relevant news channels lead to more reactions in commit messages? If so, why and how? Which channels discuss which type of CVE?

- Commit messages -- 747.3M rows
- Materialized view on commit_data, searching messages for *CVE* -- 105250 rows
- CVE database integration (https://github.com/cve-search/cve-search)
  - Provides official references where CVEs were discussed (channel)
- Crawling Twitter (https://github.com/jonbakerfish/TweetScraper) -- 376590 rows
  - Crawled for every CVE code mentioned in the materialized view

Data, structure and sources

Does attention on relevant news channels lead to more reactions in commit messages? If so, why and how? Which channels discuss which type of CVE?



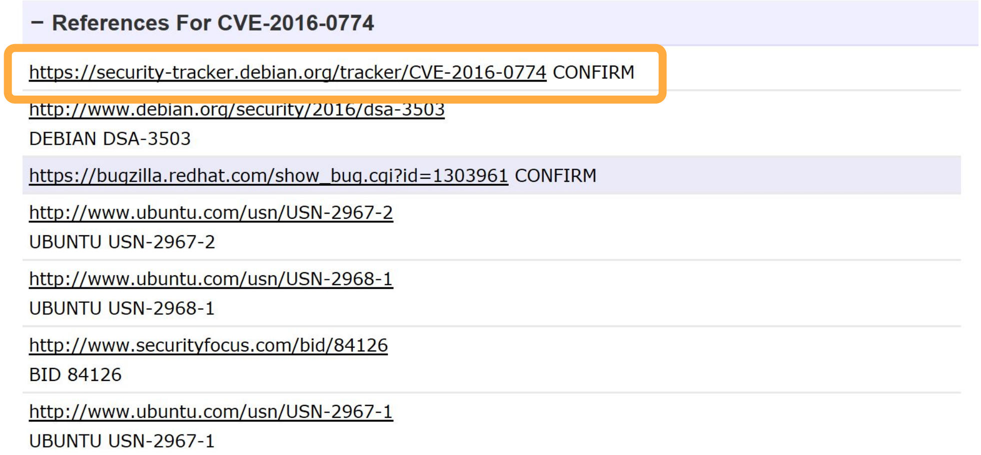
Pentest101 @Pentest101MX · 3 Apr 2016

#ITSecurity #ITSec TrendLabs Security Intelligence BlogCritical '**CVE-2015-1805**'... blog.trendmicro.com/trendlabs-secu..., see more tweetedtimes.com/Pentest101MX?s...

Critical 'CVE-2015-1805' Vulnerability Allows Perma...

On March 18, Google published a security advisory for a critical vulnerability CVE-2015-1805 that applied to rooting apps. This bug allows malicious apps to gain "r...

blog.trendmicro.com



− References For CVE-2016-0774

https://security-tracker.debian.org/tracker/CVE-2016-0774 CONFIRM

http://www.debian.org/security/2016/dsa-3503
DEBIAN DSA-3503

https://bugzilla.redhat.com/show_bug.cgi?id=1303961 CONFIRM

http://www.ubuntu.com/usn/USN-2967-2
UBUNTU USN-2967-2

http://www.ubuntu.com/usn/USN-2968-1
UBUNTU USN-2968-1

http://www.securityfocus.com/bid/84126
BID 84126

http://www.ubuntu.com/usn/USN-2967-1
UBUNTU USN-2967-1

Next steps

- Reorganise Twitter data
- Mine external data
    - Extract links contained in tweets
    - Extract links from CVE database reference entry
    - Migrate results into PostgreSQL database?
- Categorize links by type or platform
- Attempt clustering of link types and type of CVE
- Check temporal distribution of reactions to CVE announcement
- If time: Dig deeper to understand reasons behind the popularity of some CVEs
- If time: Run similar checks between the external data and Github Issues