

# **An Investigation into Vulnerability Databases**

Bachelor Thesis  
(final presentation)

31<sup>st</sup> March 2020

Brian Schweigler

# Used Terms

## **provider**

organization and people behind a vulnerability database

## **vulnerability report**

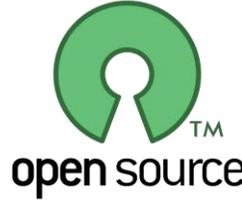
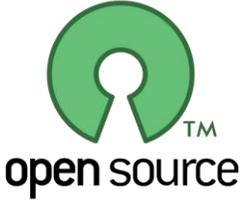
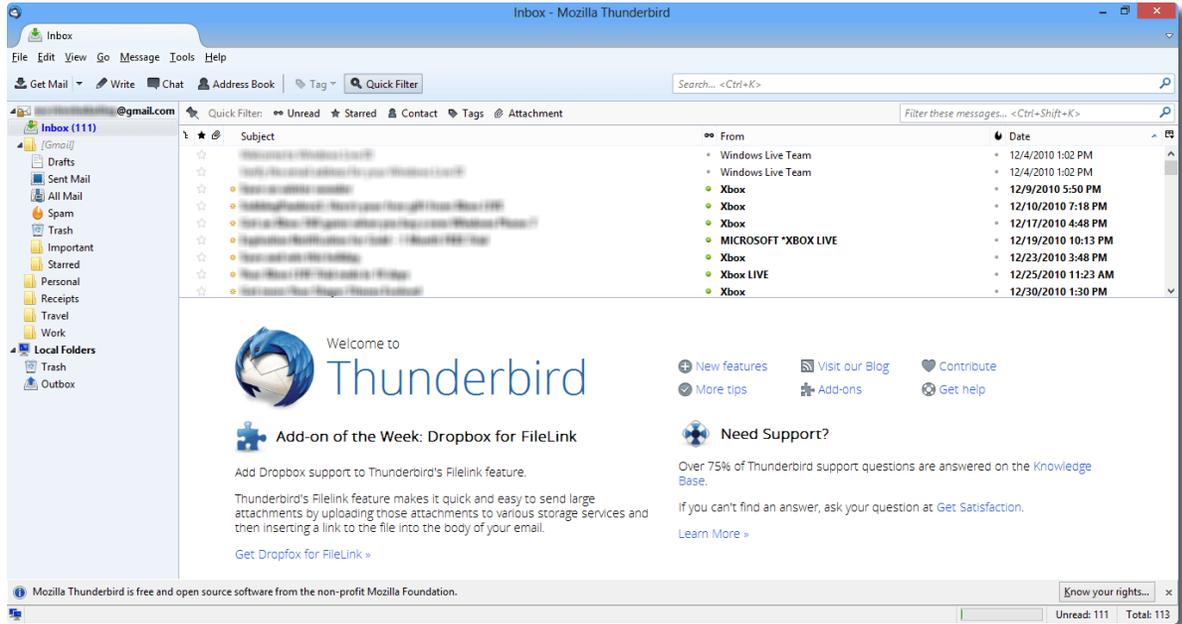
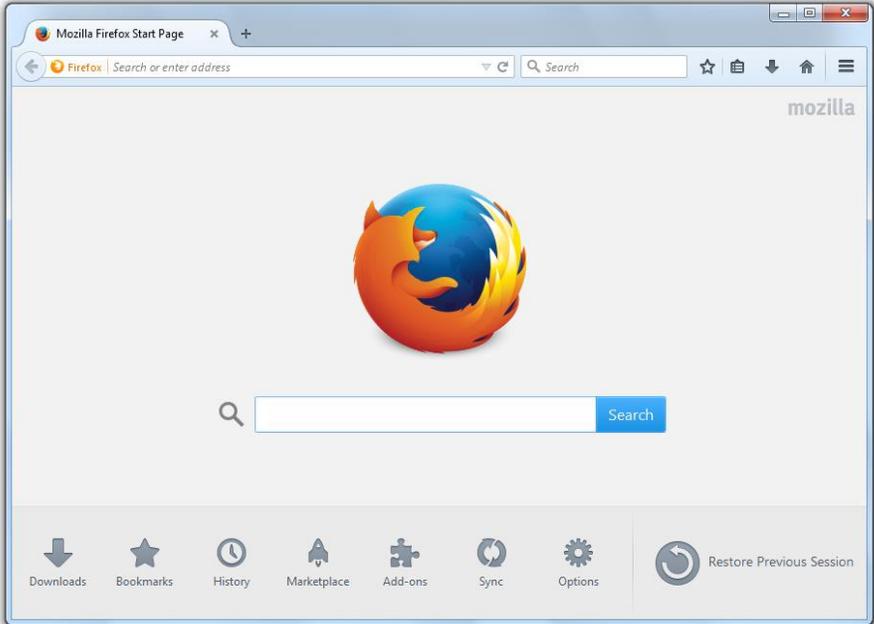
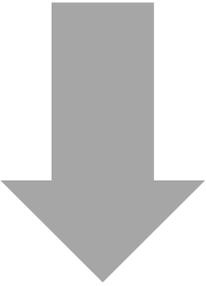
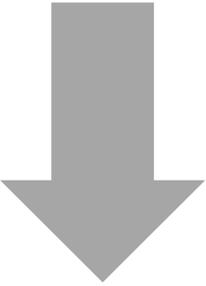
a vulnerability, an exploit, or a security-related document submitted to a provider

## **severity**

potential impact of a vulnerability

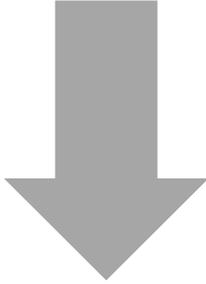
# libraries with security flaws

Gecko HTML renderer / Australis XUL GUI / ...

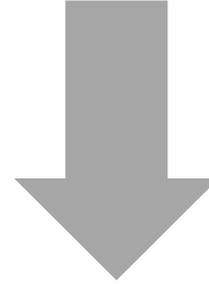


# libraries with security flaws

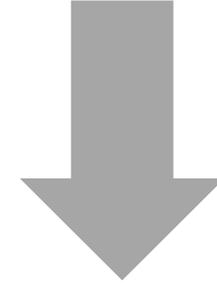
mpg123 / ffmpeg / ...



 <sup>TM</sup>  
closed source



 <sup>TM</sup>  
closed source



 <sup>TM</sup>  
closed source

... but how can we leverage vulnerability databases?

# Our Idea

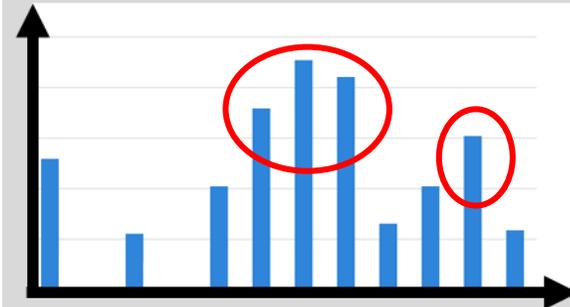
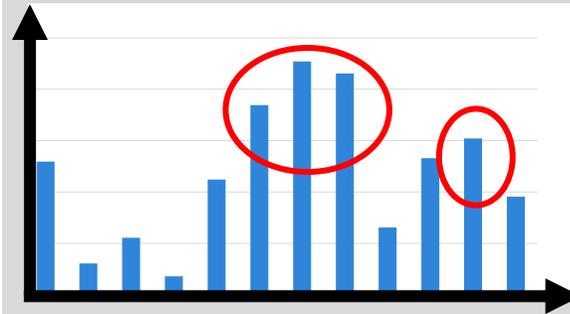
software



application



vulnerability statistics



dependency information

shares library with  
security flaws

...

...

⋮

# What Do We Need to Know?

## RQ1

What are the **affiliations** between the different vulnerability database providers, and **what data** do they collect?

First, we need data!

First, we need data!

LOTS of data.

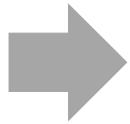
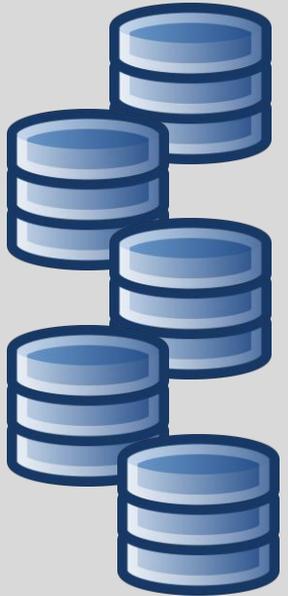
# Evaluated Providers

<b>Provider</b>	<b>Free</b>	<b>Founded in</b>	<b>Owner</b>	<b>Source</b>	<b>Remarks</b>
CVE	yes	1999	Mitre	open	simple, unbiased overview
Exploit-DB	yes	2004	Offensive Security	open	exploit and app download
NVD	yes	2000	NIST	closed	lists affected software
RAPID7	no	2000	RAPID7	closed	evaluates package use
Snyk	yes	2006	Snyk Ltd.	closed	evaluates package use

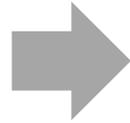
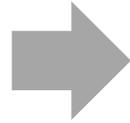
# Data Collection



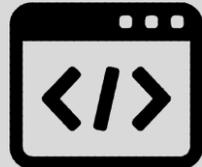
vendor  
db



servers

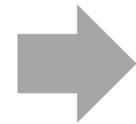


interface



Kotlin code

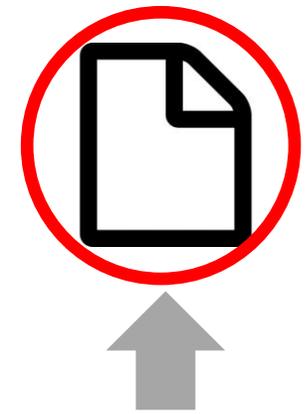
web-scraping &  
regex matching



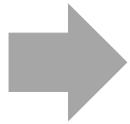
unified  
db



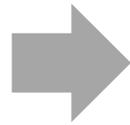
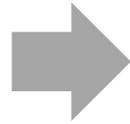
# Data Collection



vendor db



servers

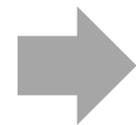


interface



Kotlin code

web-scraping &  
regex matching



unified db



# Data Processing

Python scripts applied to vulnerability descriptions for ...

tokenizing strings

counting word frequencies in descriptions

grouping by dates

removing stop-words

plotting (140+)



Exploit-DB requires a valid HTTP header for requests  
User-Agent: <product> / <product-version>

Exploit-DB requires regular IP address changes

character set encoding issues  
(who needs special characters?!)

# Our Dataset

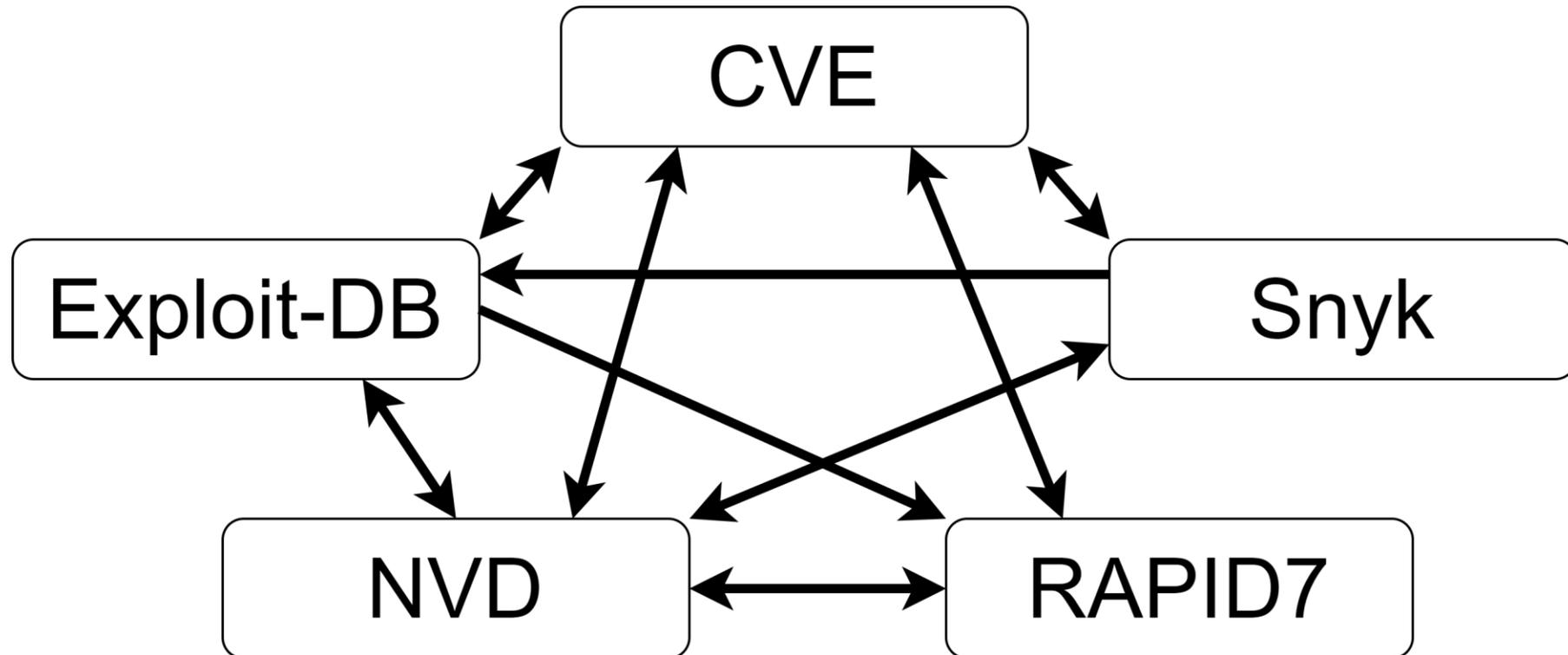
<b>Provider</b>	<b>First Report</b>	<b>Cut-off date</b>	<b># Entries</b>
CVE	1999	12.12.2019	156,828
NVD	1999	03.12.2019	133,477
Exploit-DB	1988	01.11.2019	44,539
RAPID7	2004	18.12.2019	30,849
Snyk	2005	17.11.2019	4,605

# What Do We Need to Know?

## RQ1

What are the **affiliations** between the different vulnerability database providers, and **what data** do they collect?

# Vulnerability Provider Affiliations



# Supported Feature Sets

Provider	ID	CWE	Author	Title	Description	Reference	Date	Score	Special
CVE	✓	✗	✗	✗	✓	✓	✓	✗	✓
NVD	✓	✓	✗	✗	✓	✓	✓	✓	✓
Exploit-DB	✓	✗	✓	✓	✗	✗	✓	✗	✓
RAPID7	✓	✗	✗	✓	✓	✓	✓	✓	✓
Snyk	✓	✓	✗	✓	✓	✓	✓	✓	✓

# What Do We Need to Know?

## RQ1

What are the **affiliations** between the different vulnerability database providers, and **what data** do they collect?

## RQ2

Are the **vulnerability scoring systems** used consistently?

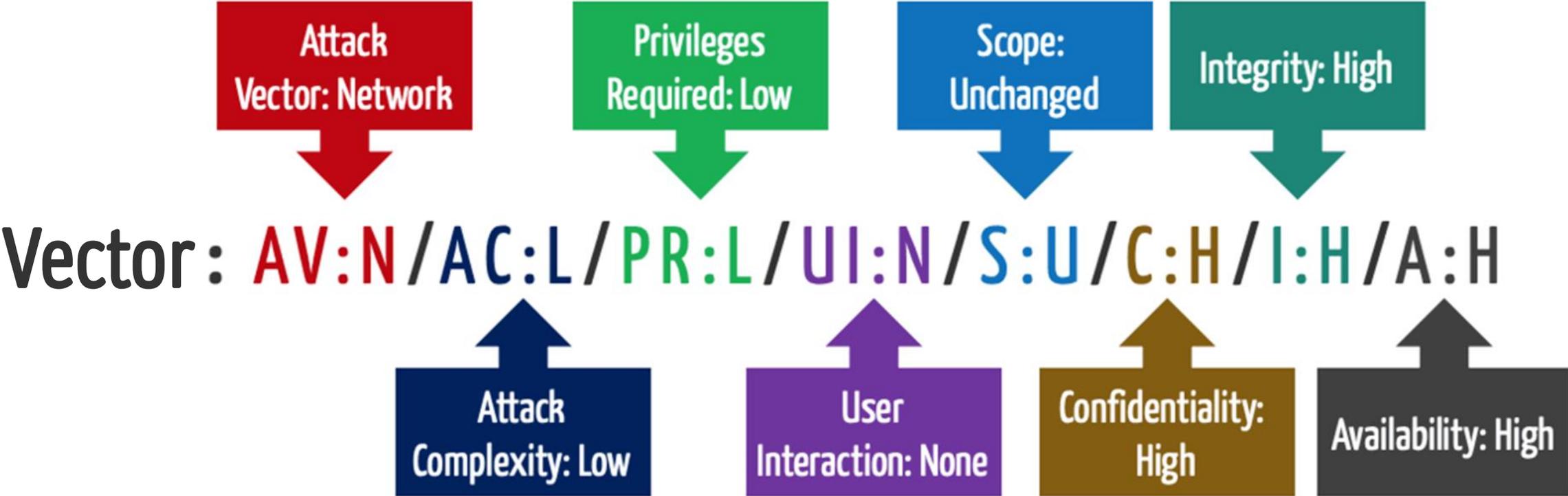
# Scoring

“The Common Vulnerability Scoring Standard (CVSS) provides a way to capture principal characteristics of a vulnerability and produce a numerical score [1-10] reflecting its severity.”

— FIRST (Forum of Incident Response and Security Teams)

# CVSS Details

multiple versions are in use (CVSS V2, CVSS V3, CVSS V3.1)



# Scoring

Provider	CVSS V2	CVSS V3	V2 Average	V3 Average
CVE			n/a	n/a
NVD			<b>6.2</b>	<b>7.4</b>
Exploit-DB			n/a	n/a
RAPID7			<b>6.3</b>	n/a
Snyk			n/a	<b>6.6</b>

# What Do We Need to Know?

## RQ1

What are the **affiliations** between the different vulnerability database providers, and **what data** do they collect?

## RQ2

Are the **vulnerability scoring systems** used consistently?

## RQ3

Can we **see trends** based on the collected data?

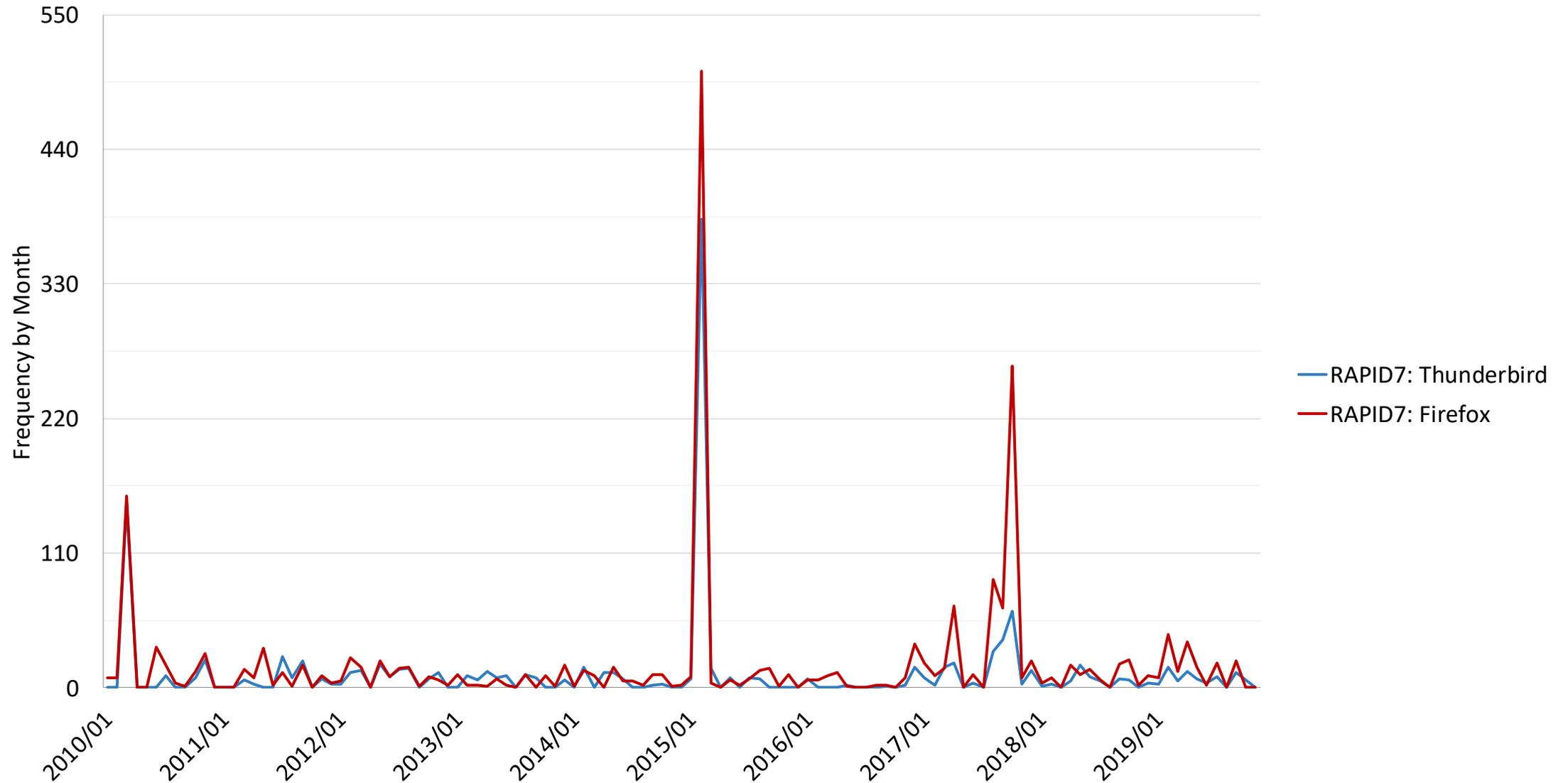
# Determination of Affected Software

200 most frequent non-stop words found in the descriptions:  
(filtered by software names)

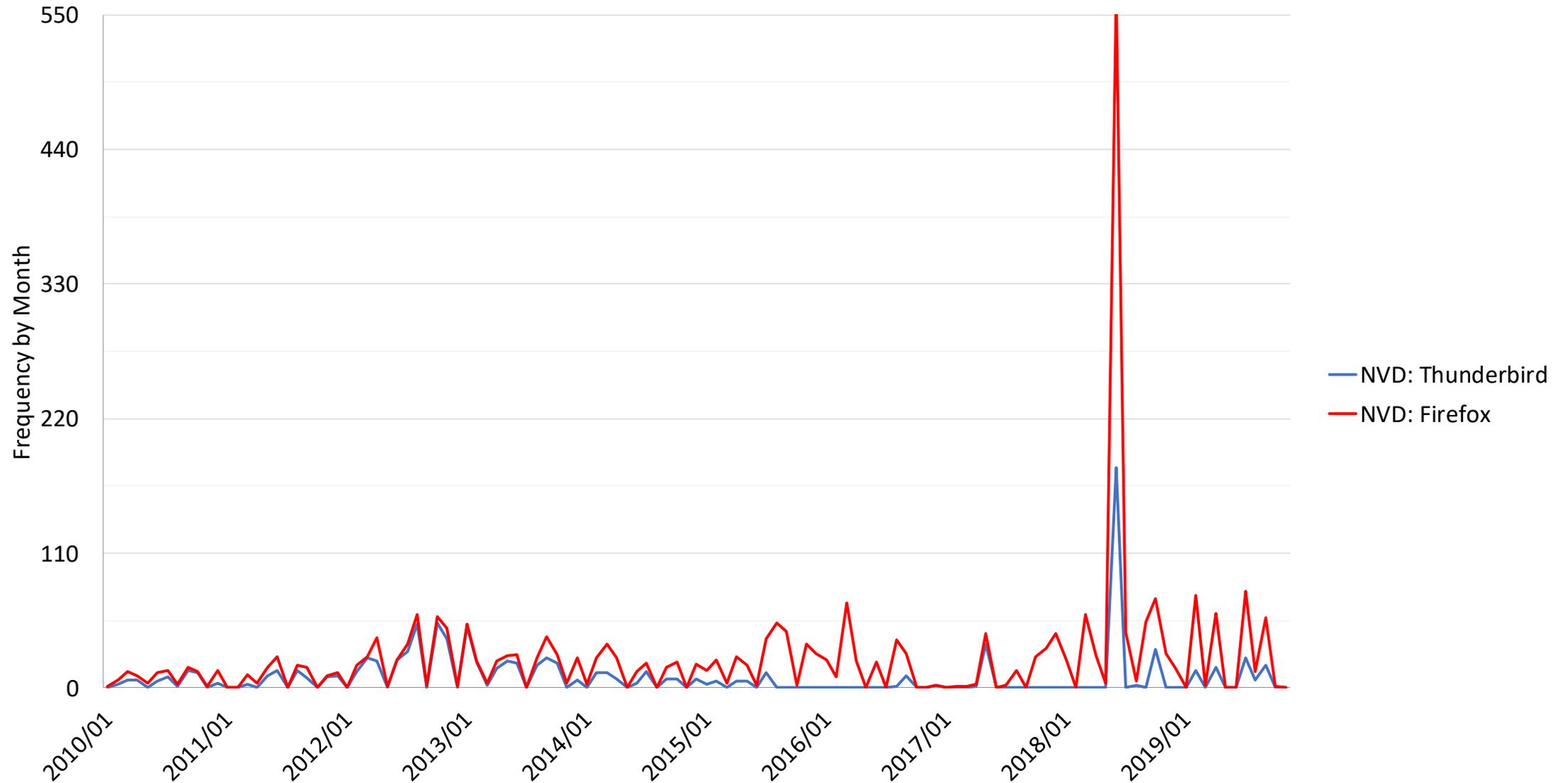
Acrobat, Android, Apache, Apple, Chrome,  
Chromium\*, FFmpeg\*, Flash, Firefox, Java,  
Linux, Thunderbird, Windows, Wordpress

... and then we created some plots!

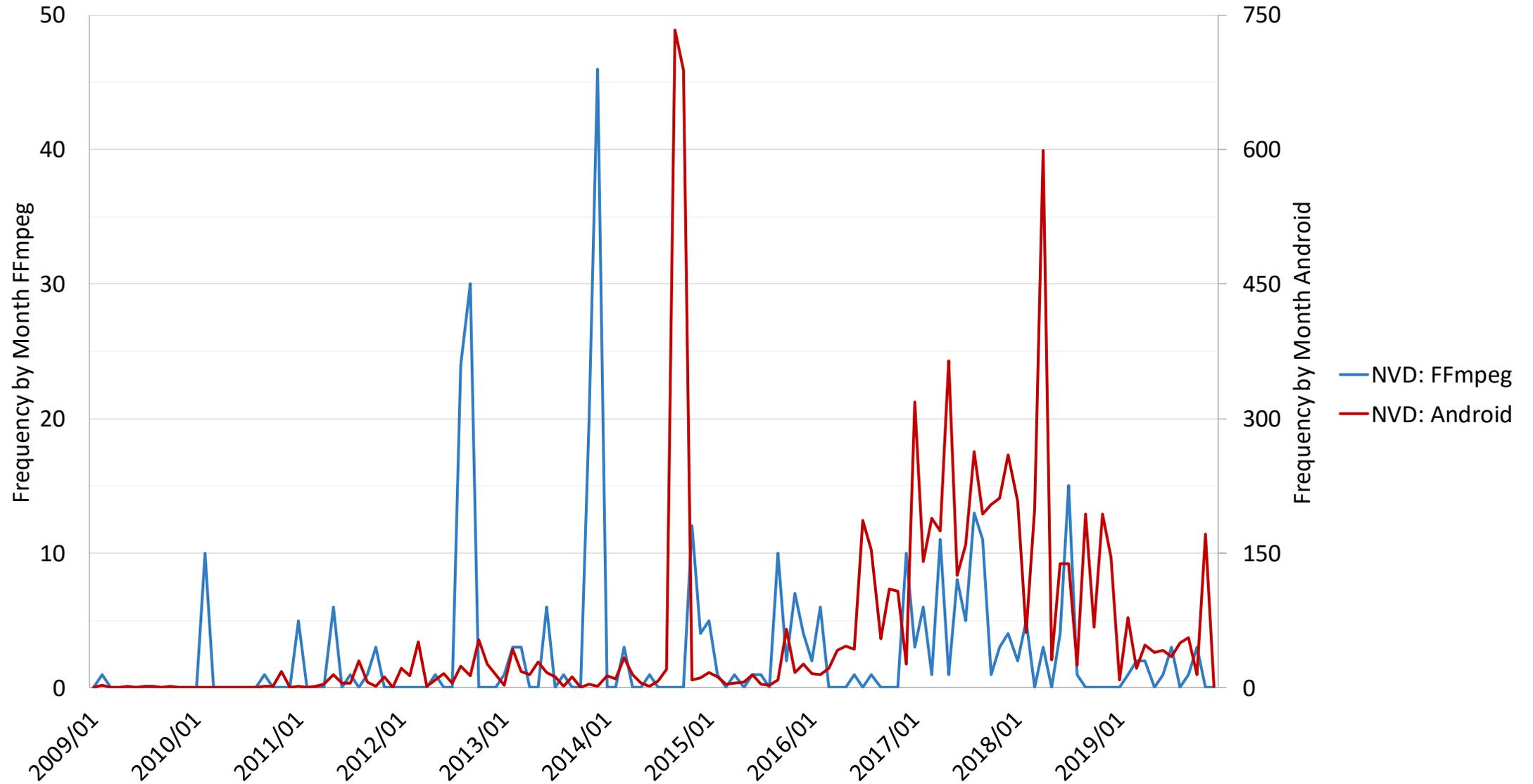
# Trends - Shared Code (Thunderbird / Firefox)



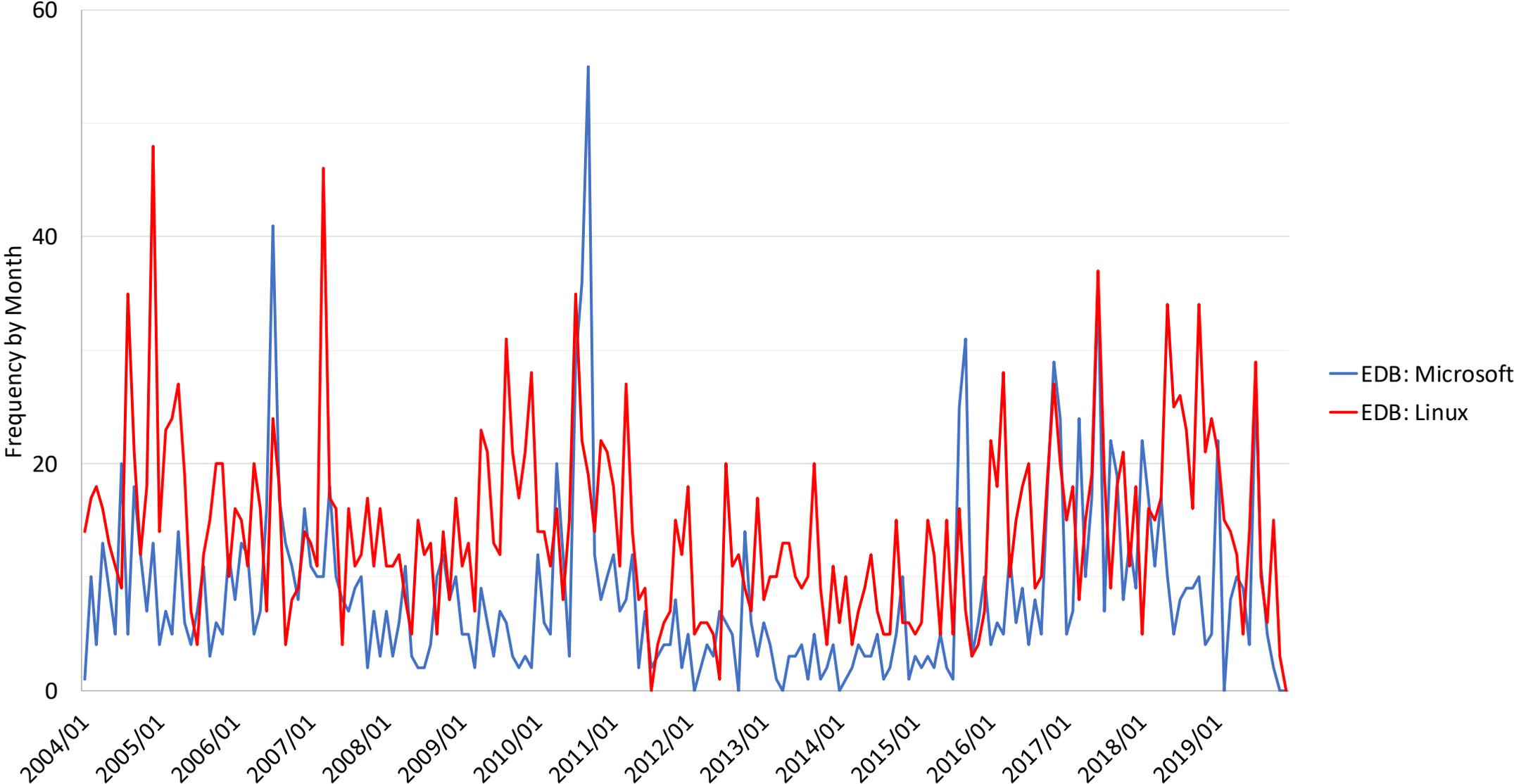
# Trends - Shared Code (Thunderbird / Firefox)



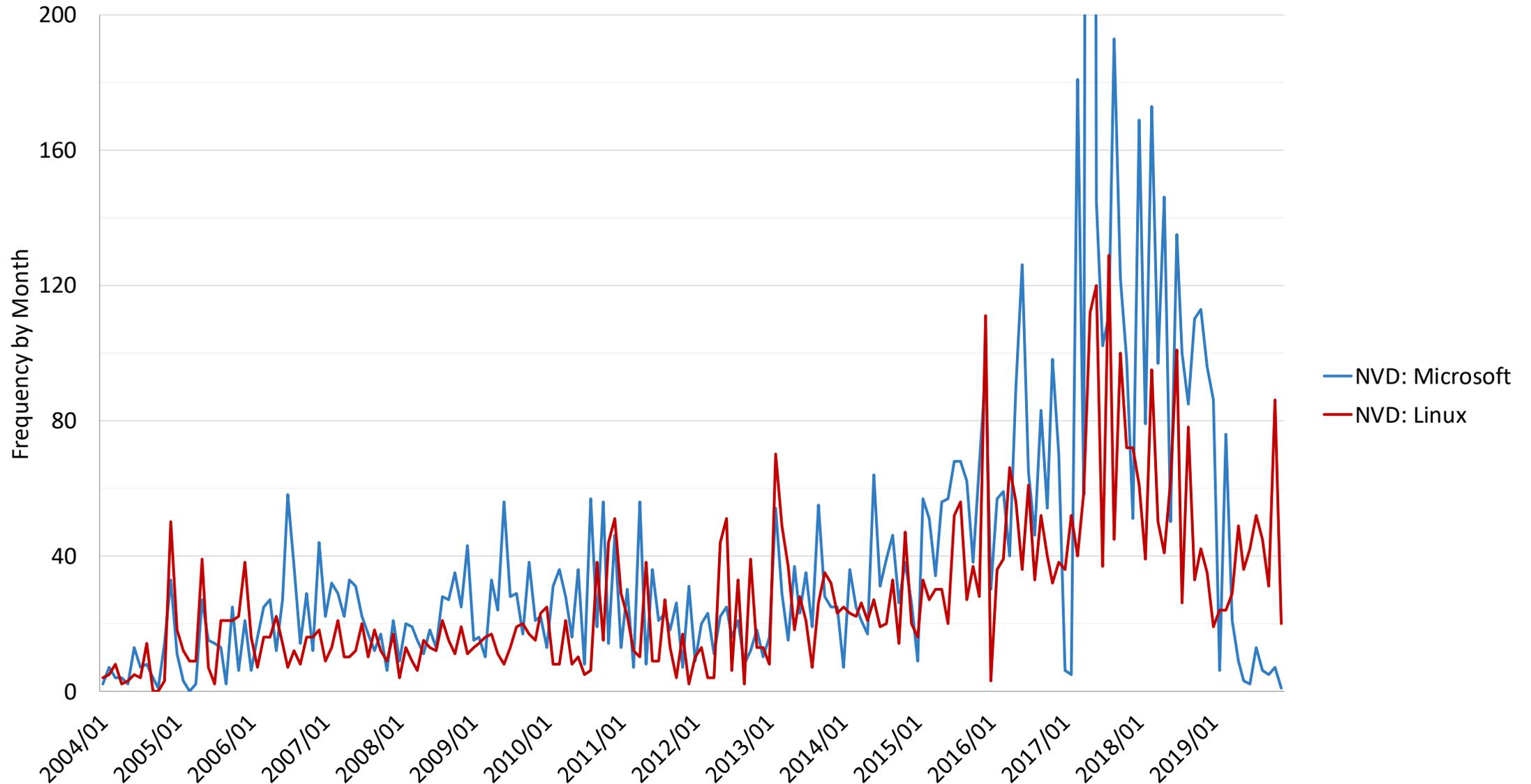
# Trends - Shared Code (FFmpeg / Android)



# Trends - Related Software (Microsoft / Linux)



# Trends - Related Software (Microsoft / Linux)

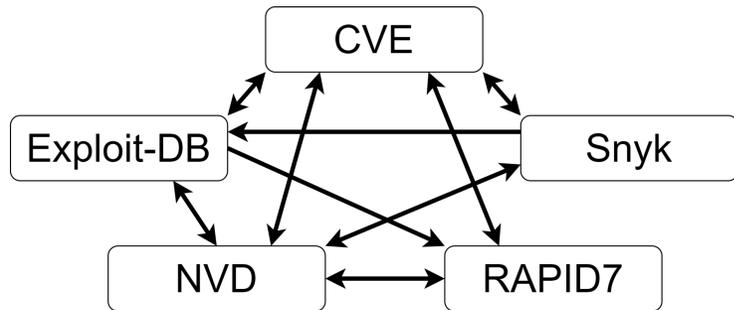


# Future Work

machine learning to find corresponding patterns

improved NLP on descriptions

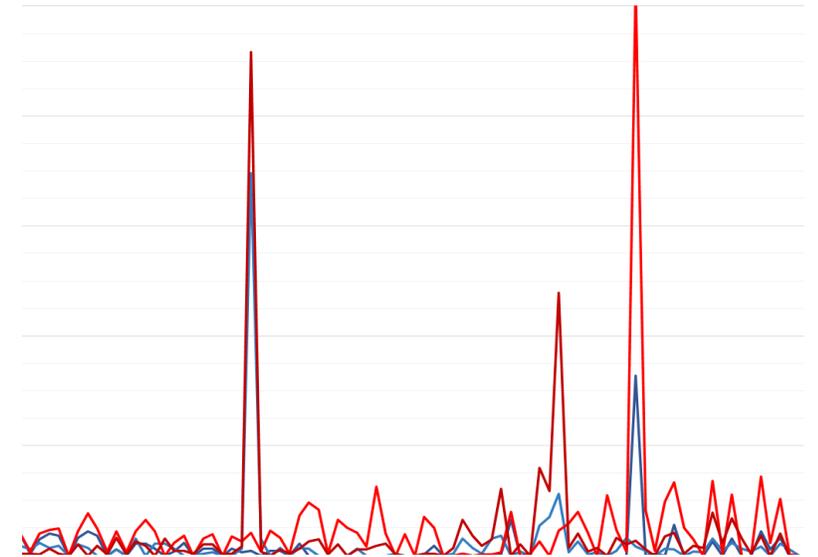
# Summary



**RQ1:**  
**Affiliations & Data**

Provider	CVSS V2	CVSS V3	V2 Average	V3 Average
CVE	✗	✗	n/a	n/a
NVD	✓	✓	6.2	7.4
Exploit-DB	✗	✗	n/a	n/a
RAPID7	✓	✗	6.3	n/a
Snyk	✗	✓	n/a	6.6

**RQ2:**  
**Scoring**



**RQ3:**  
**Trends**