INTERNATIONAL WORKSHOP ON SECURE SOFTWARE ENGINEERING IN DEVOPS AND AGILE DEVELOPMENT

SECBENCH

A Database of Real Security Vulnerabilities

Speaker: Rui Abreu

Authors: Sofia Reis[1] & Rui Abreu[2]

[1] *Faculty of Engineering of University of Porto, Portugal*

[2] *IST, University of Lisbon & INESC-ID, Portugal*

Oslo, Norway

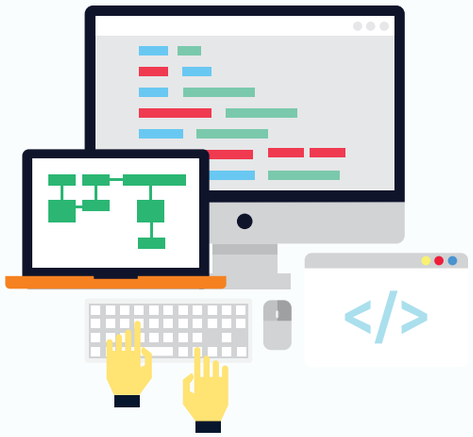
September, 2017

OVERVIEW

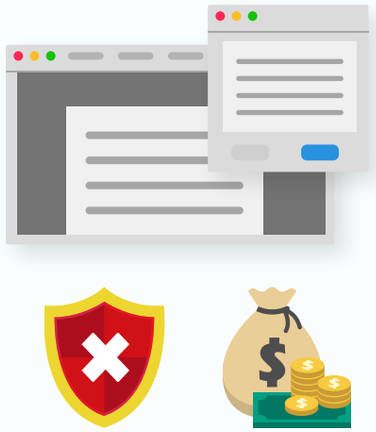
1. **Introduction**
    1. Problem
    2. Motivation
    3. Research Questions
2. **Related Work**
3. **Extracting and Isolating Vulnerabilities from Github Repositories**
    1. Patterns
    2. Test Cases Structure
    3. Vulnerability Diagnosis
4. **Empirical Evaluation**
    1. Database
    2. Research Questions
5. **Conclusions and Future Work**

PROBLEM

Complex Software

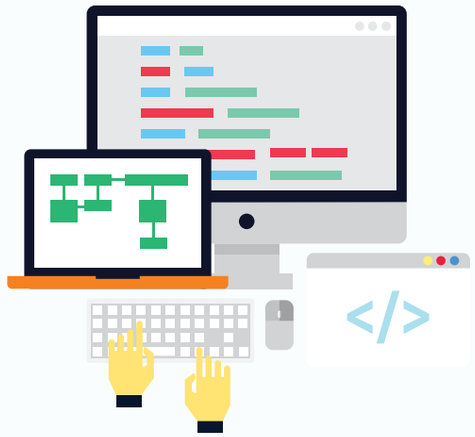


PROBLEM

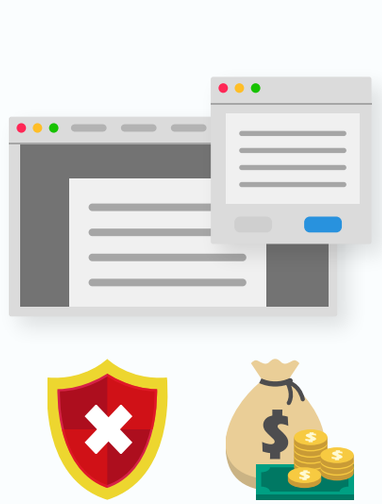


High costs associated with the identification and correction of defects

Complex Software



PROBLEM



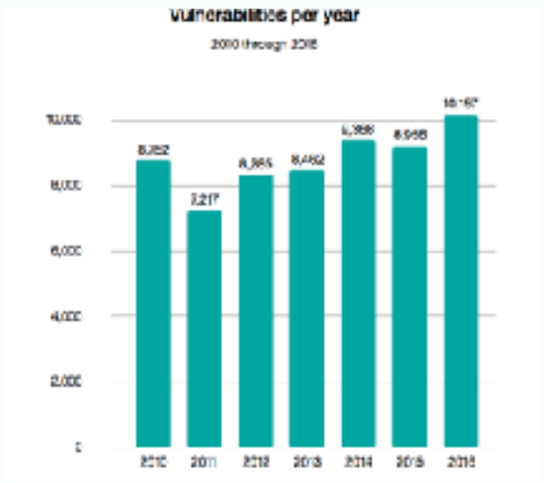


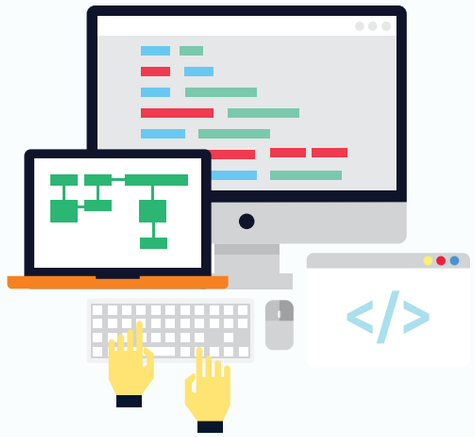
Fig.1 - IBM's X-Force Threat Intelligence 2017

High costs associated with the identification and correction of defects

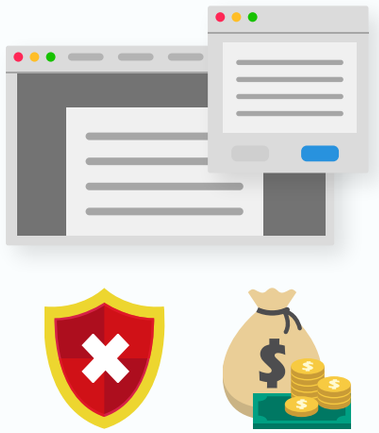
+

Increase of the number of security vulnerabilities (last 6 years)

Complex Software



PROBLEM



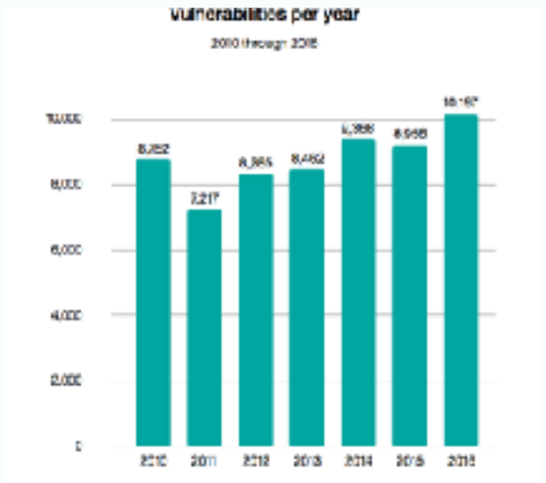
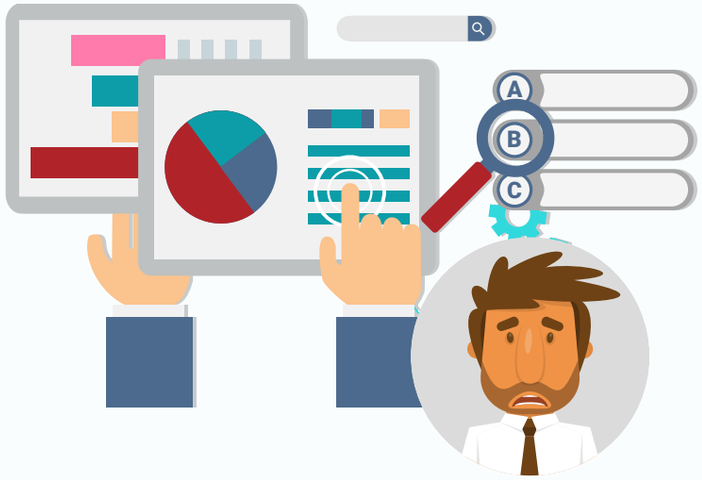


Fig.1 - IBM's X-Force Threat Intelligence 2017

High costs associated with the identification and correction of defects

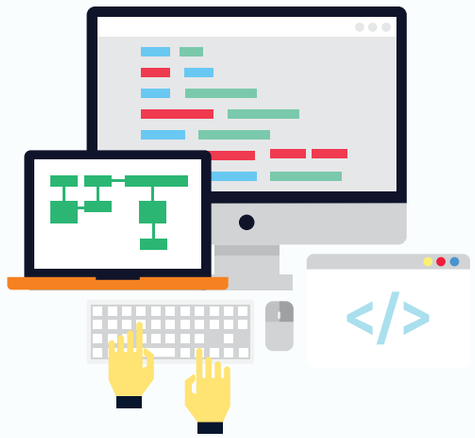
+

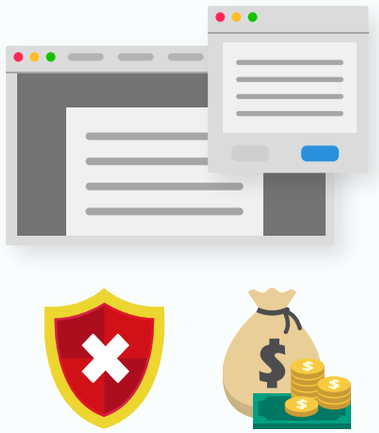
Increase of the number of security vulnerabilities (last 6 years)



Static Analysis Tools (SATs) far from being **effective** and **efficient**

Complex Software





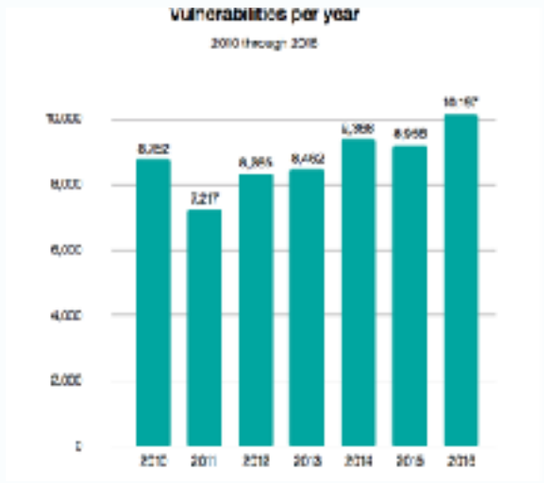


Fig.1 - IBM's X-Force Threat Intelligence 2017

High costs associated with the identification and correction of defects

+

Increase of the number of security vulnerabilities (last 6 years)

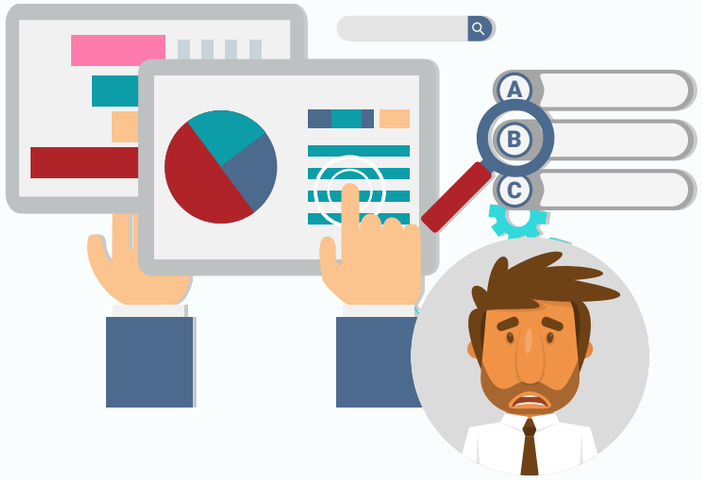
PROBLEM

Lack of databases

containing real test cases

Sources

- Github
- Bitbucket
- SourceForge
- BugZilla
- Travis CI



Static Analysis Tools (SATs) far from being **effective** and **efficient**

MOTIVATION

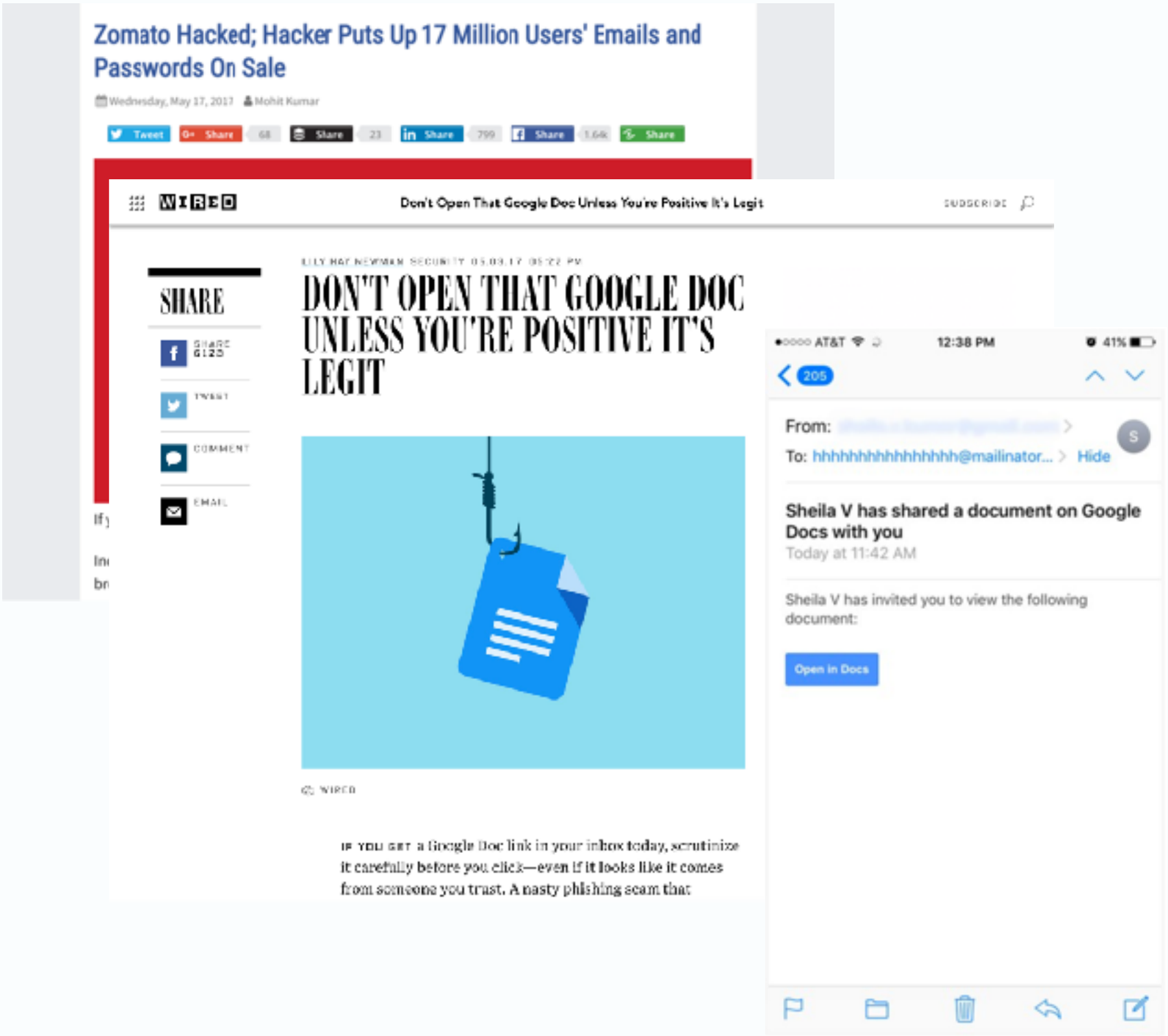
Zomato Hacked; Hacker Puts Up 17 Million Users' Emails and Passwords On Sale

Wednesday, May 17, 2017 Mohit Kumar

If you ever ordered food from Zomato, You should be Worried!

India's largest online restaurant guice Zomato confirmed today that the company has suffered a data breach and that accounts details of millions of its users have been stolen from its database.

MOTIVATION



Zomato Hacked; Hacker Puts Up 17 Million Users' Emails and Passwords On Sale

Wednesday, May 17, 2017 Mohit Kumar

Tweet Share 68 Share 23 Share 799 Share 1.6k Share

WIRED

Don't Open That Google Doc Unless You're Positive It's Legit

SUBSCRIBE

SHARE

SHARE 6125

TWEET

COMMENT

EMAIL

LILY HAY NEWMAN SECURITY 05.03.17 05:22 PM

DON'T OPEN THAT GOOGLE DOC UNLESS YOU'RE POSITIVE IT'S LEGIT

WIRED

IF YOU GET a Google Doc link in your inbox today, scrutinize it carefully before you click—even if it looks like it comes from someone you trust. A nasty phishing scam that

AT&T 12:38 PM 41%

205

From:

To: hhhhhhhhhhhhhhhhh@mailinator... Hide

S

Sheila V has shared a document on Google Docs with you

Today at 11:42 AM

Sheila V has invited you to view the following document:

Open in Docs

MOTIVATION

Zomato Hacked; Hacker Puts Up 17 Million Users' Emails and Passwords On Sale

Wednesday, May 17, 2017 Mohit Kumar

Tweet Share 68 Share 23 Share 799 Share 1.64k Share

WIRED Don't Open That Google Doc Unless You're Positive

SHARE

SHARE 6125

TWEET

COMMENT

EMAIL

LILY HAY NEWMAN SECURITY 05.03.17 05:22 PM

DON'T OPEN THAT GOOGLE D
UNLESS YOU'RE POSITIVE IT'
LEGIT

WIRED

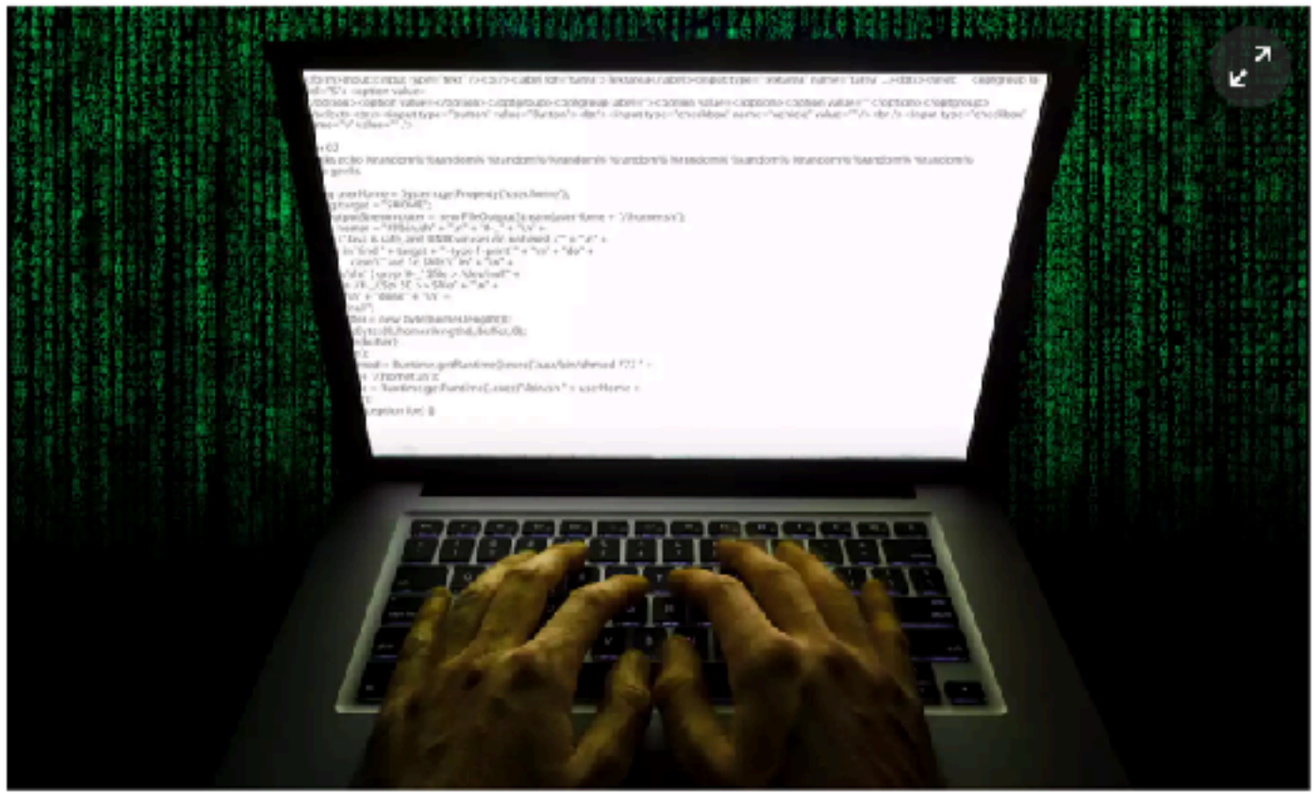
IF YOU GET a Google Doc link in your inbox today, scrut
it carefully before you click—even if it looks like it cor
from someone you trust. A nasty phishing scam that

Hacking

DDoS attack that disrupted internet was largest of its kind in history, experts say

Dyn, the victim of last week's denial of service attack, said it was orchestrated using a weapon called the Mirai botnet as the 'primary source of malicious attack'

Major cyber attack disrupts internet service across Europe and US



MOTIVATION


Wana Decrypt0r 2.0
Ooops, your files have been encrypted!
English
What Happened to My Computer?
Your important files are encrypted.
Many of your documents, photos, videos, databases and other files are no longer accessible because they have been encrypted. Maybe you are busy looking for a way to recover your files, but do not waste your time. Nobody can recover your files without our decryption service.
Can I Recover My Files?
Sure. We guarantee that you can recover all your files safely and easily. But you have not so enough time.
You can decrypt some of your files for free. Try now by clicking <Decrypt>.
But if you want to decrypt all your files, you need to pay.
You only have 3 days to submit the payment. After that the price will be doubled.
Also, if you don't pay in 7 days, you won't be able to recover your files forever.
We will have free events for users who are so poor that they couldn't pay in 6 months.
How Do I Pay?
Payment is accepted in Bitcoin only. For more information, click <About bitcoin>.
Please check the current price of Bitcoin and buy some bitcoins. For more information, click <How to buy bitcoins>.
And send the correct amount to the address specified in this window.
After your payment, click <Check Payment>. Best time to check: 9:00am - 11:00am
Payment will be raised on
5/16/2017 00:47:55
Time Left
02:23:57:37
Your files will be lost on
5/20/2017 00:47:55
Time Left
06:23:57:37
About bitcoin
How to buy bitcoins?
Contact Us
bitcoin ACCEPTED HERE
Send $300 worth of bitcoin to this address:
Copy
Check Payment
Decrypt

RESEARCH QUESTIONS

RQ1: *Is there enough information available on open-source repositories to create a database of software security vulnerabilities?*

RQ1: *What are the most prevalent security patterns on open-source repositories?*

2. RELATED WORK

Databases

- Software-artifact Infrastructure Repository (SIR)
- Juliest Test Suites
- CodeChecker
- OWASP Benchmark
- Defects4j
- Safety-db

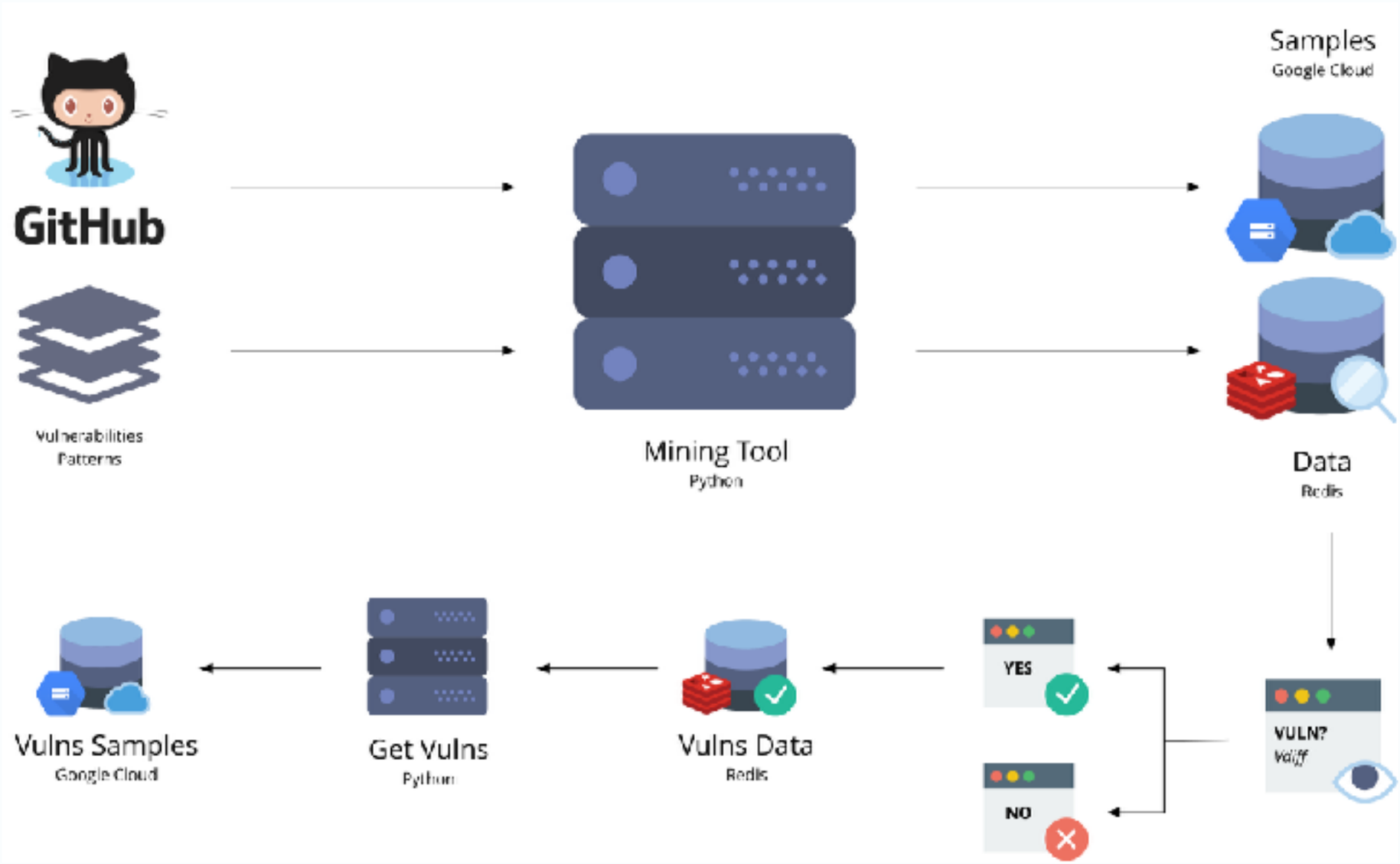
2. RELATED WORK

Databases

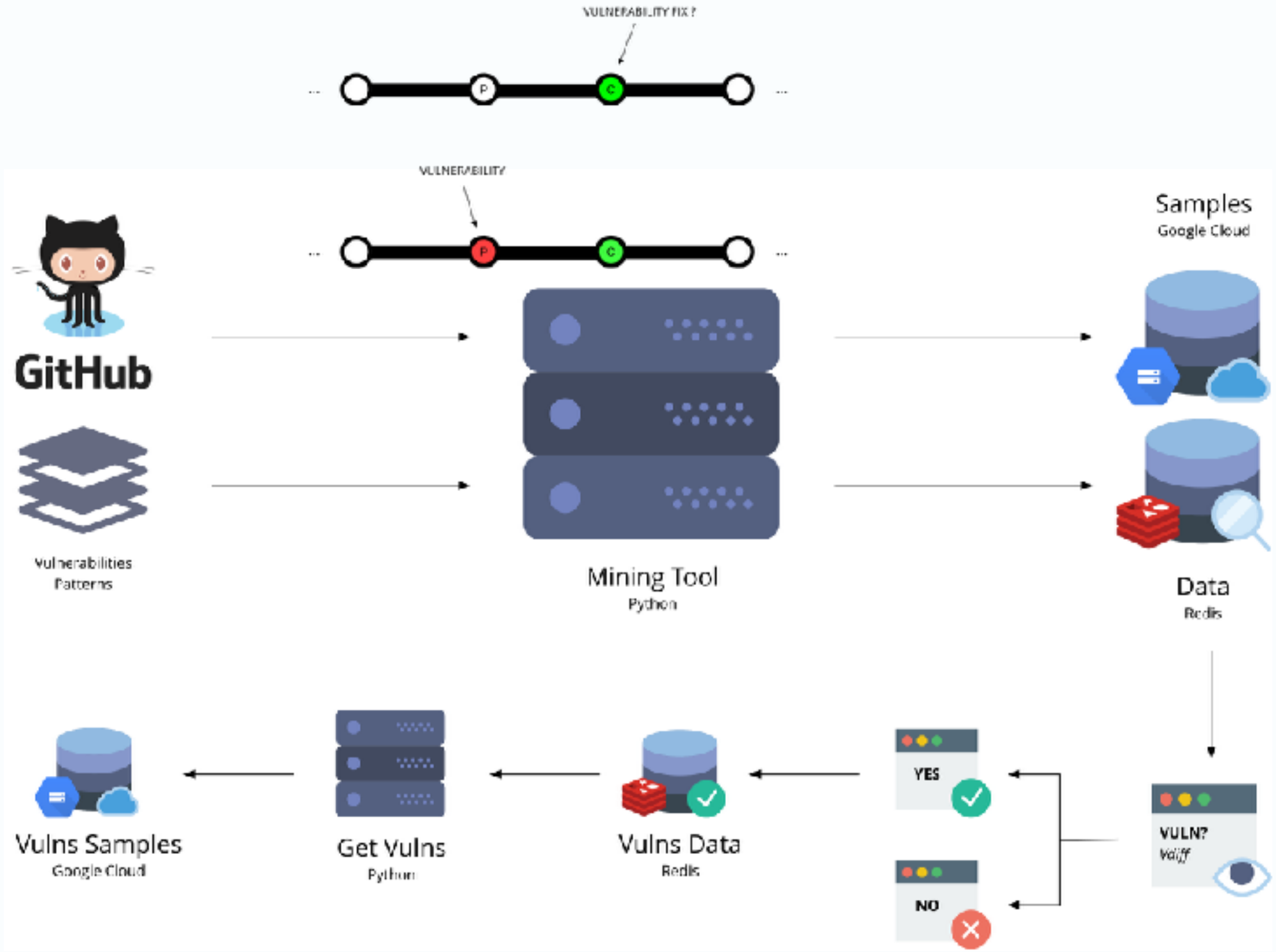
- Software-artifact Infrastructure Repository (SIR)
- Juliest Test Suites
- CodeChecker
- OWASP Benchmark
- Defects4j
- Safety-db

1 benchmark has only real security vulnerabilities

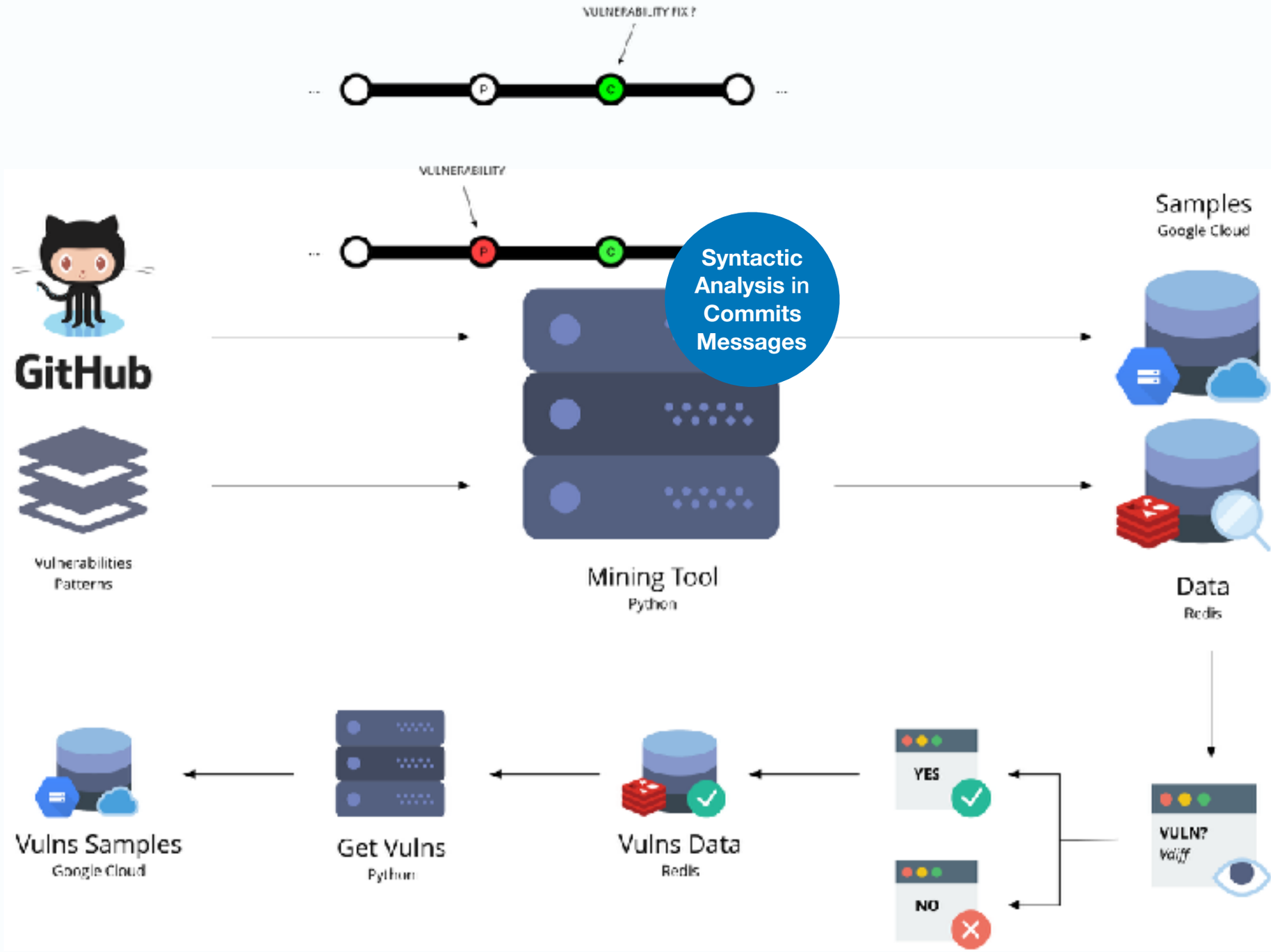
3. EXTRACTING AND ISOLATING VULNERABILITIES FROM GITHUB



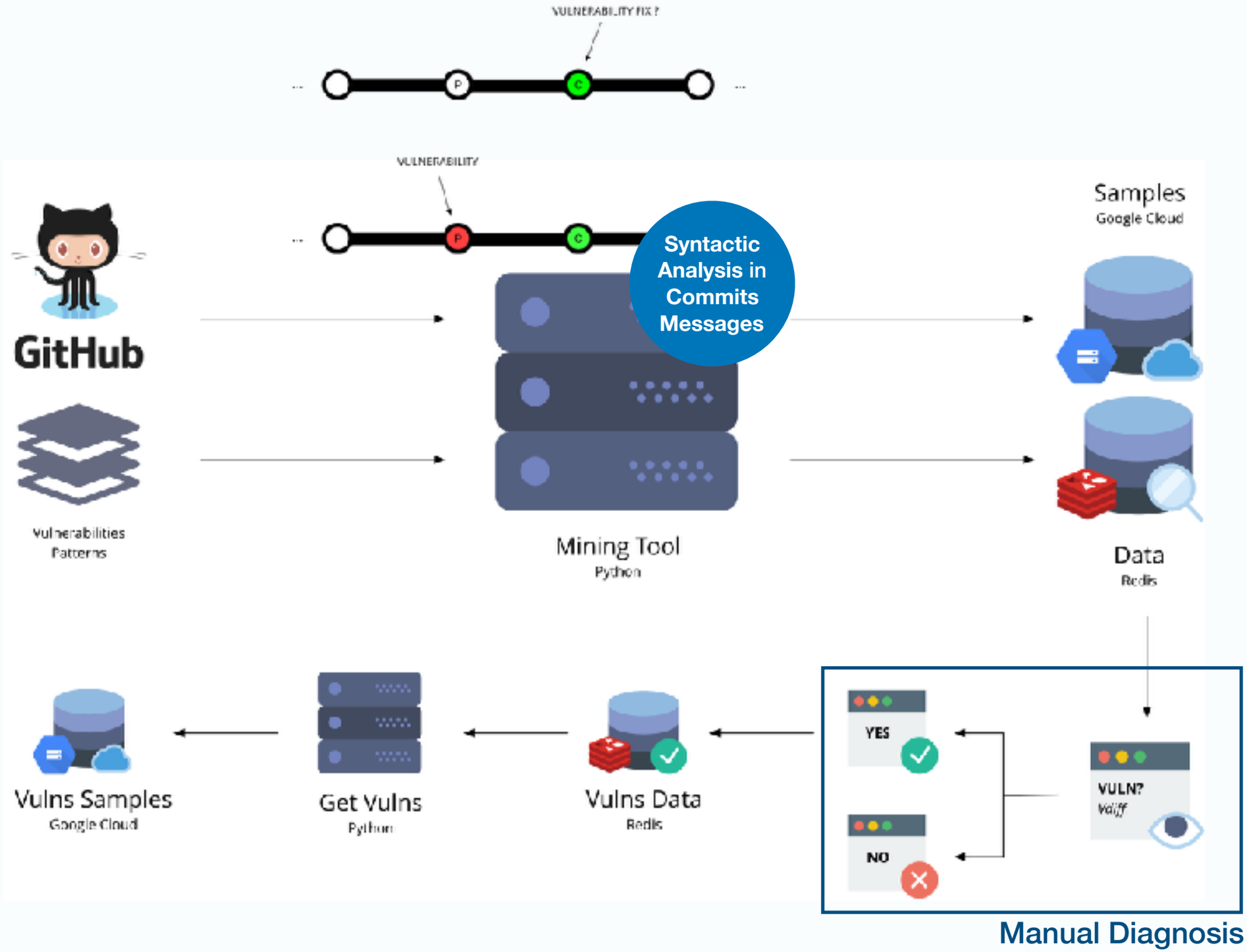
3. EXTRACTING AND ISOLATING VULNERABILITIES FROM GITHUB



3. EXTRACTING AND ISOLATING VULNERABILITIES FROM GITHUB



3. EXTRACTING AND ISOLATING VULNERABILITIES FROM GITHUB

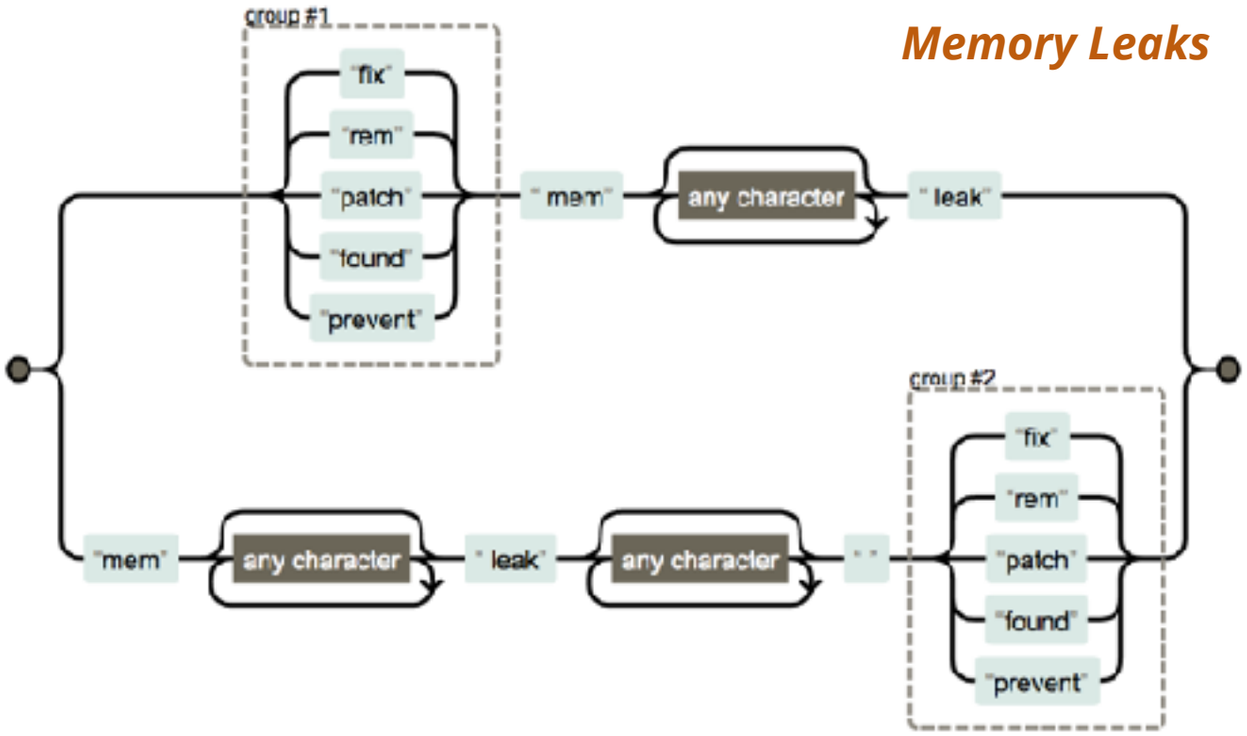


PATTERNS

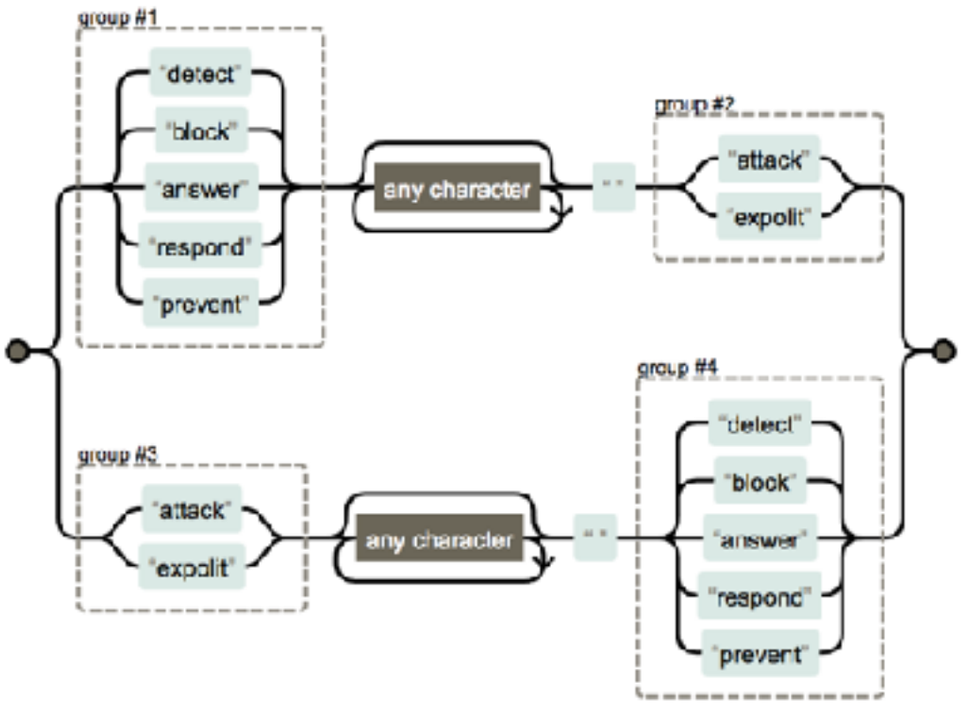
TOP 10 OWASP, Memory Leaks, Resource Leaks, Overflow, Denial-of-Service and more

Total of 17 patterns created

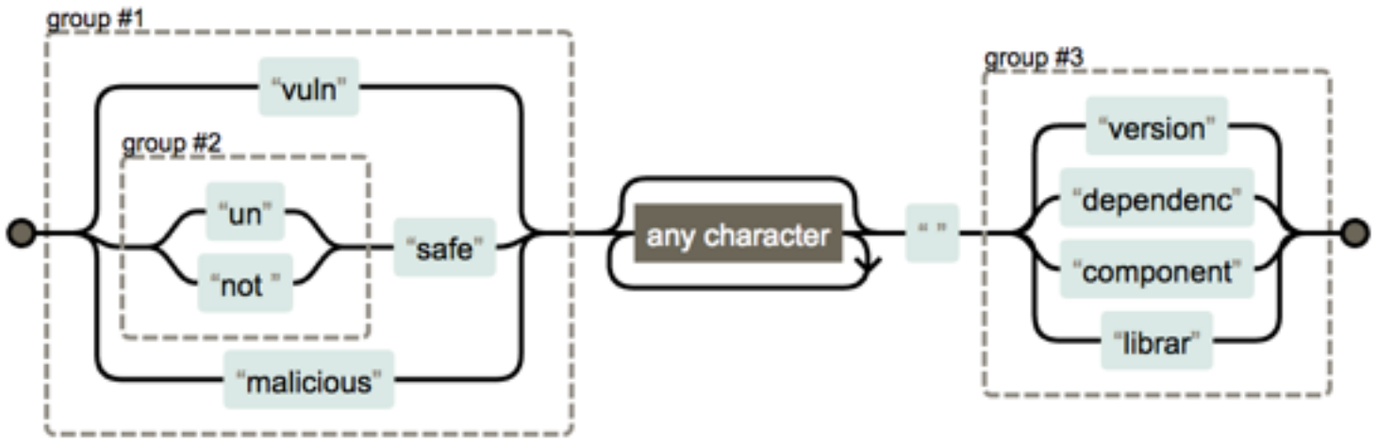
Memory Leaks



A6 - Insufficient Attack Protection



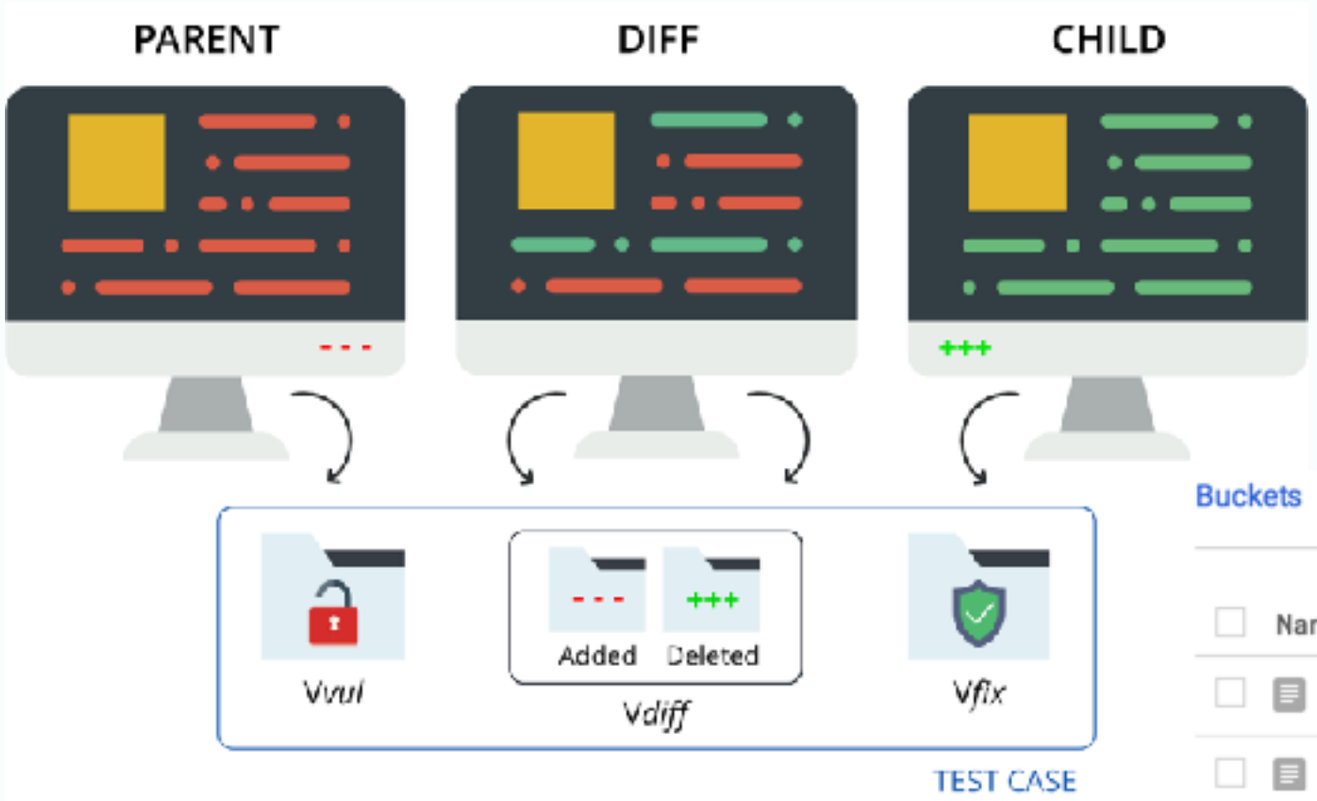
A9 - Using Components with Known Vulnerabilities

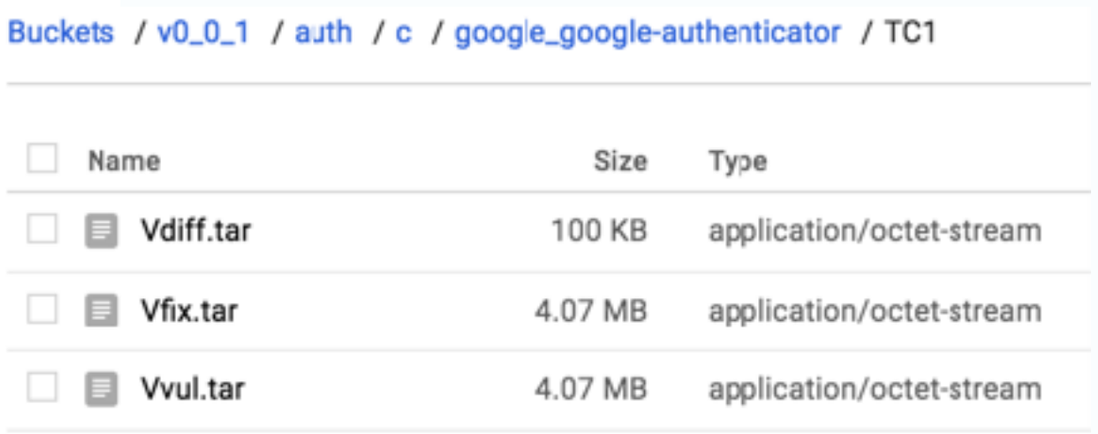


TEST CASES STRUCTURE

Juliet Test Cases

Vvul - **Vulnerable** source code from the parent commit.
Vfix - **Non-vulnerable** source code from the commit where the pattern was caught.
Vdiff - Two folders, containing the **added lines to fix the vulnerability** and the **deleted lines representing the security vulnerability**.





HOW ARE VULNERABILITIES IDENTIFIED?

AUTOMATED *IN COMMITS MESSAGES AND* ***MANUALLY*** *IN SOURCE CODE*

Example 1

staging: fbtft: Fix buffer overflow vulnerability

Browse files

Module copies a user supplied string (mocule parameter) into a buffer
using strncpy() and does not check that the buffer is null terminated.

Replace call to strncpy() with call to strlcpy() ensuring that the
buffer is null terminated.

Signed-off-by: Tobin C. Harding <me@tobir.cc>
Signed-off-by: Greg Kroah-Hartman <gregkh@linuxfouncation.org>

master v4.12 v4.12-rc1

tcharding committed with gregkh on Feb 15 — 1 parent ca5af1f commit 8414fe1ba2ff2422c4b73bbef3b035a1562c88a

Showing 1 changed file with 1 addition and 1 deletion. Unified Split

2 drivers/staging/fbtft/fbtft_device.c View

@@ -1483,7 +1483,7 @@ static int __init fbtft_device_init(void)

```
-                       strncpy(displays[i].spi->modalias, name, SPI_NAME_SIZE);
+                       strlcpy(displays[i].spi->modalias, name, SPI_NAME_SIZE);
```

Buffer Overflow

Example 2

```
+ sock = socket(sa->sa_family, SOCK_STREAM, 0); // SOCKET INITIALIZATION
  if (0 > sock) {
          zlog(ZLOG_SYSERROR, "failed to create new listening socket: socket()");
          return -1;
  }

  setsockopt(sock, SOL_SOCKET, SO_REUSEADDR, &flags, sizeof(flags));

  if (wp->listen_address_domain == FPM_AF_UNIX) {
          if (fpm_socket_unix_test_connect((struct sockaddr_un *)sa, socklen) == 0) {
                  zlog(ZLOG_ERROR, "An another FPM instance seems to already listen on %s", ((struct sockaddr_un *) sa)->sun_path);
                  // SOCKET NEEDS TO BE CLOSED BEFORE RETURN
                  return -1;
          }
          unlink( ((struct sockaddr_un *) sa)->sun_path);
          saved_umask = umask(0777 ^ wp->socket_mode);
  }

  if (0 > bind(sock, sa, socklen)) {
          zlog(ZLOG_SYSERROR, "unable to bind listening socket for address '%s'", wp->config->listen_address);
          if (wp->listen_address_domain == FPM_AF_UNIX) {
                  umask(saved_umask);
          }
          // SOCKET NEEDS TO BE CLOSED BEFORE RETURN
          return -1;
  }
```

```
  sock = socket(sa->sa_family, SOCK_STREAM, 0);

  if (0 > sock) {
          zlog(ZLOG_SYSERROR, "failed to create new listening socket: socket()");
          return -1;
  }

  setsockopt(sock, SOL_SOCKET, SO_REUSEADDR, &flags, sizeof(flags));

  if (wp->listen_address_domain == FPM_AF_UNIX) {
          if (fpm_socket_unix_test_connect((struct sockaddr_un *)sa, socklen) == 0) {
                  zlog(ZLOG_ERROR, "An another FPM instance seems to already listen on %s", ((struct sockaddr_un *) sa)->sun_path);
+                 close(sock);
                  return -1;
          }
          unlink( ((struct sockaddr_un *) sa)->sun_path);
          saved_umask = umask(0777 ^ wp->socket_mode);
  }

  if (0 > bind(sock, sa, socklen)) {
          zlog(ZLOG_SYSERROR, "unable to bind listening socket for address '%s'", wp->config->listen_address);
          if (wp->listen_address_domain == FPM_AF_UNIX) {
                  umask(saved_umask);
          }
+         close(sock);
          return -1;
  }
```

Resource Leaks

EMPIRICAL EVALUATION: SAMPLE CHARACTERISTICS

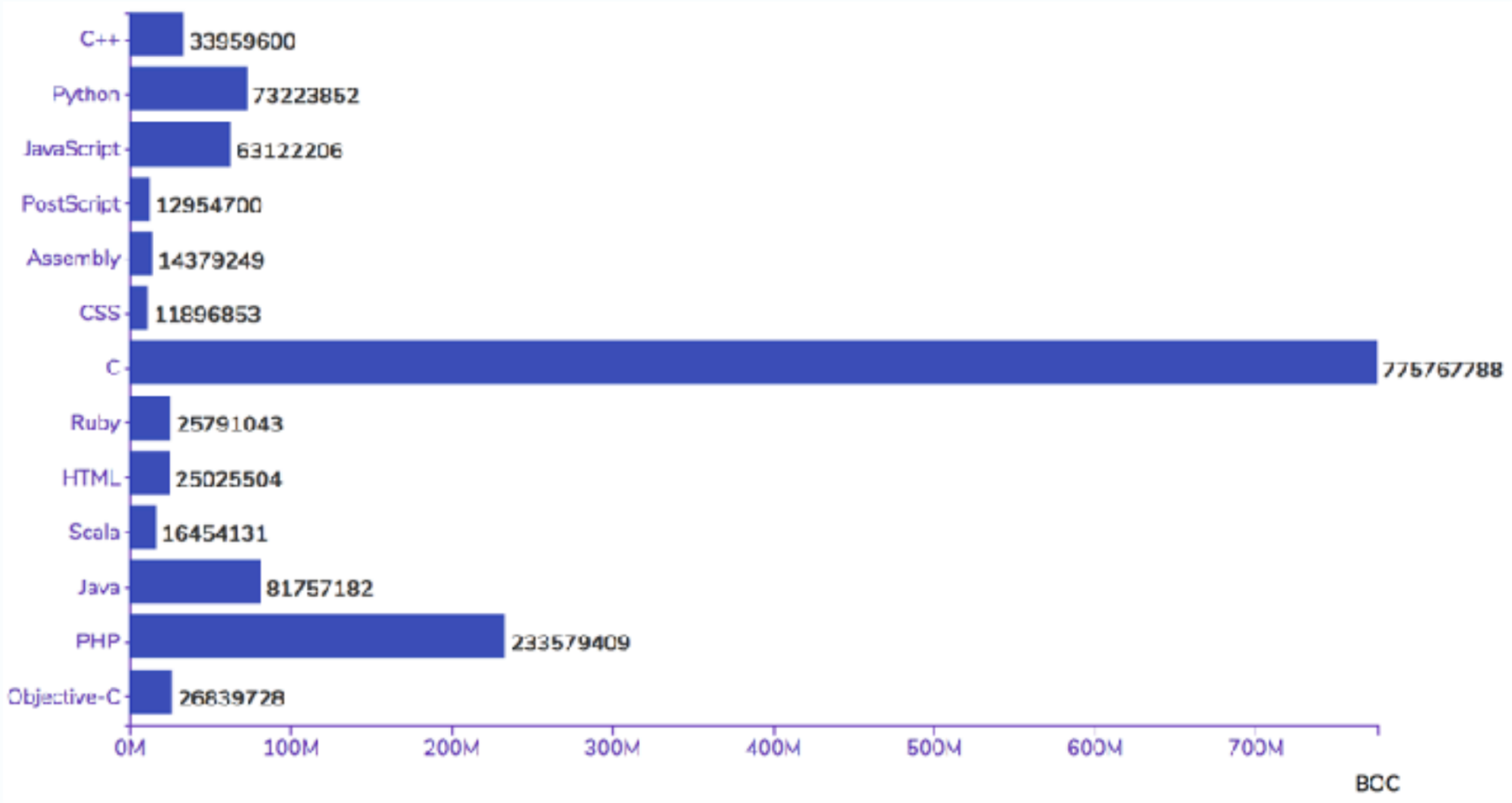
248 repositories covering considerably **9 of the top of most programming** languages

(total of 94 programming languages + sizes between 2 and 700K commits)

EMPIRICAL EVALUATION: SAMPLE CHARACTERISTICS

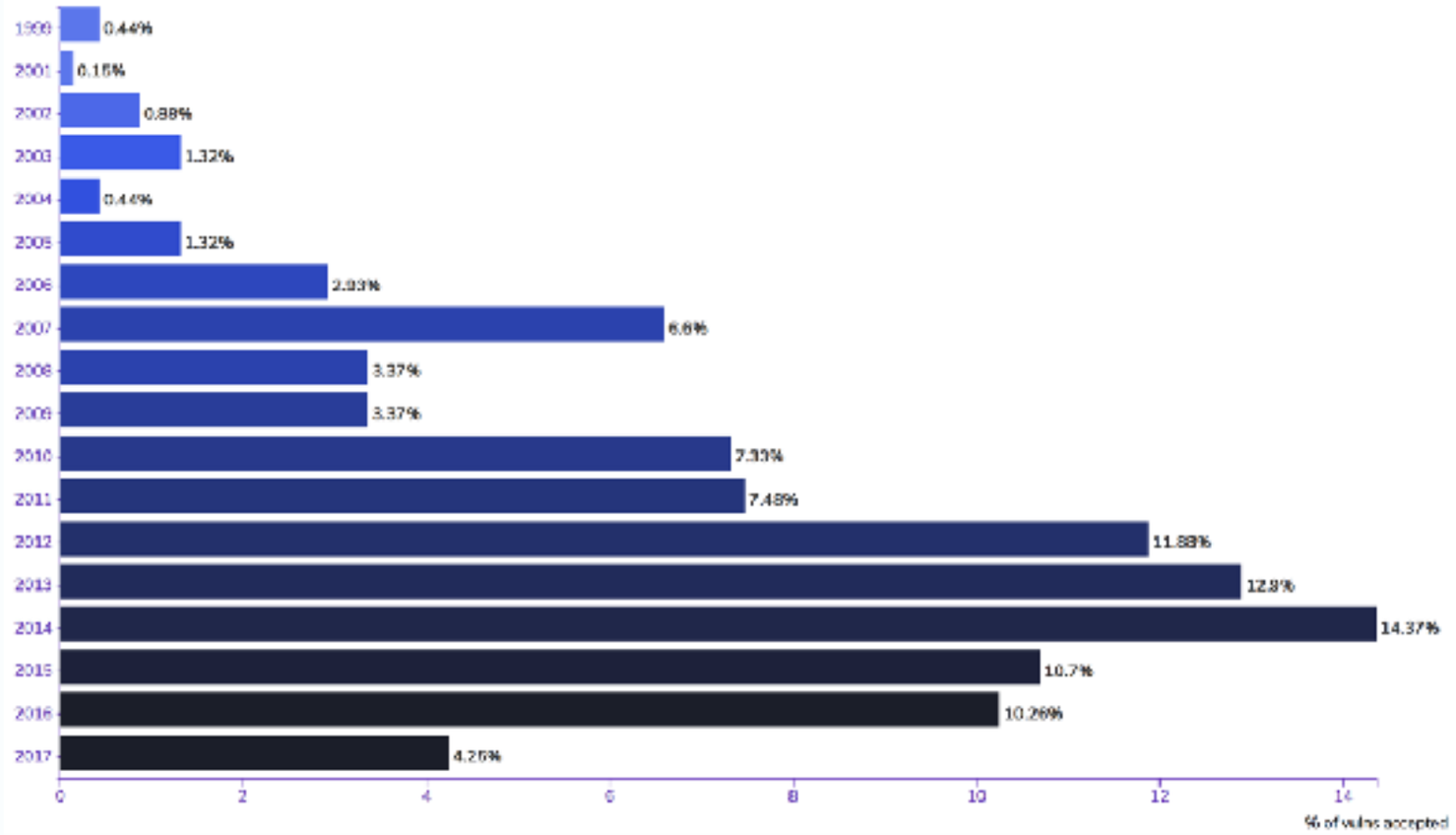
248 repositories covering considerably **9 of the top of most programming** languages

(total of 94 programming languages + sizes between 2 and 700K commits)



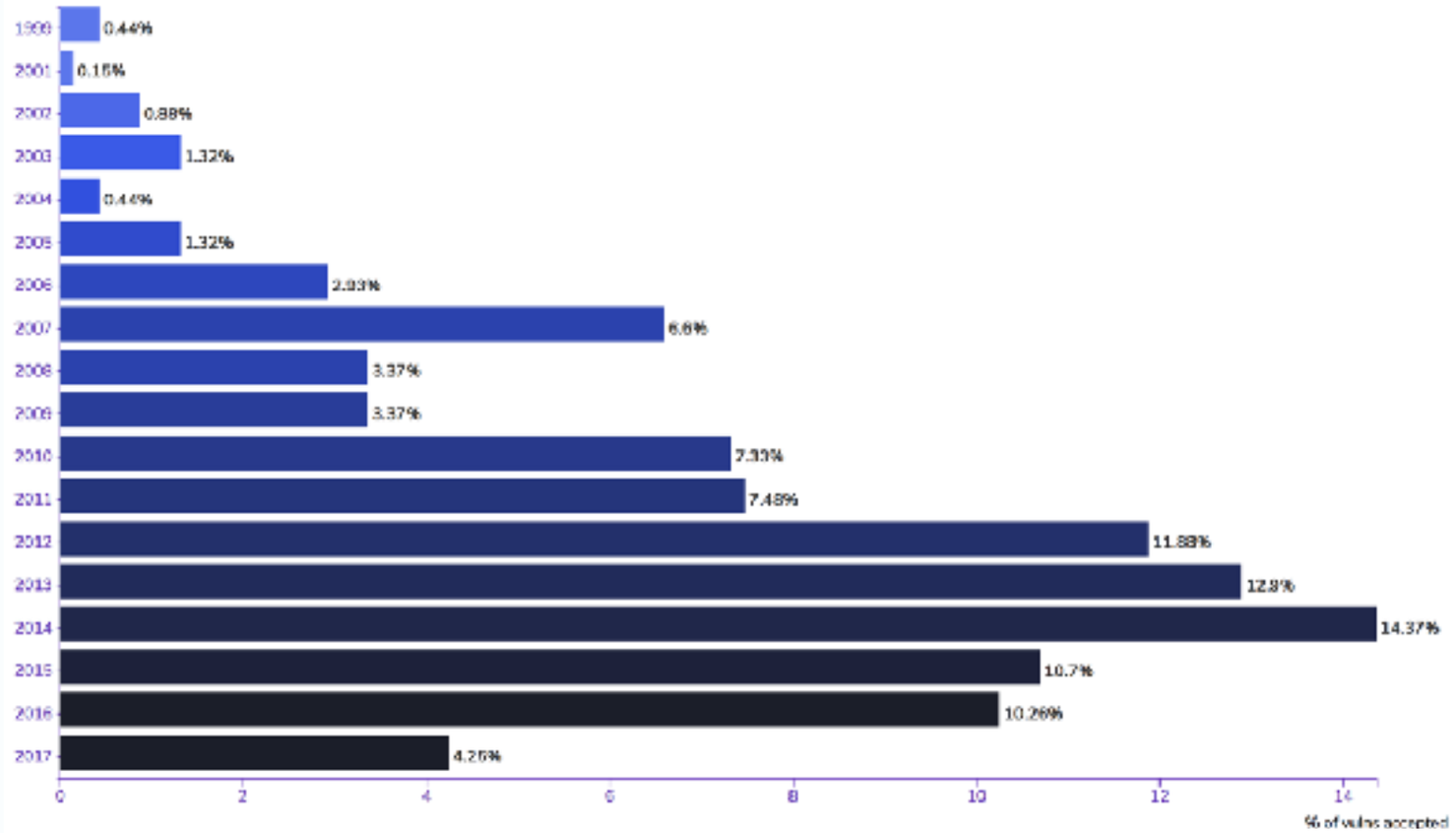
- 16/17 patterns satisfied
- 682 security vulnerabilities accepted
- Almost 6K commits manually evaluated

DATABASE OF REAL SECURITY VULNERABILITIES



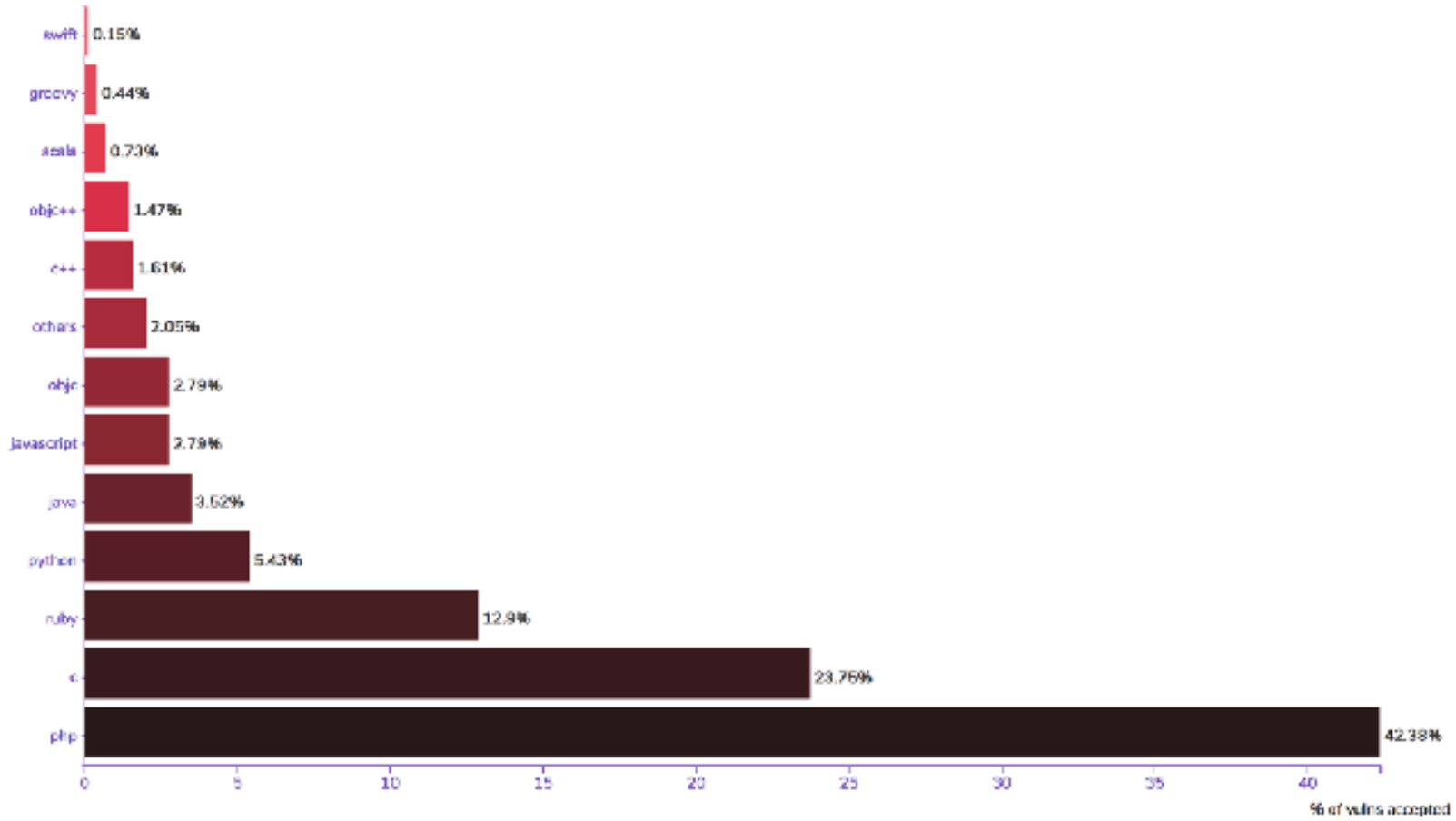
Accepted vulnerabilities per year

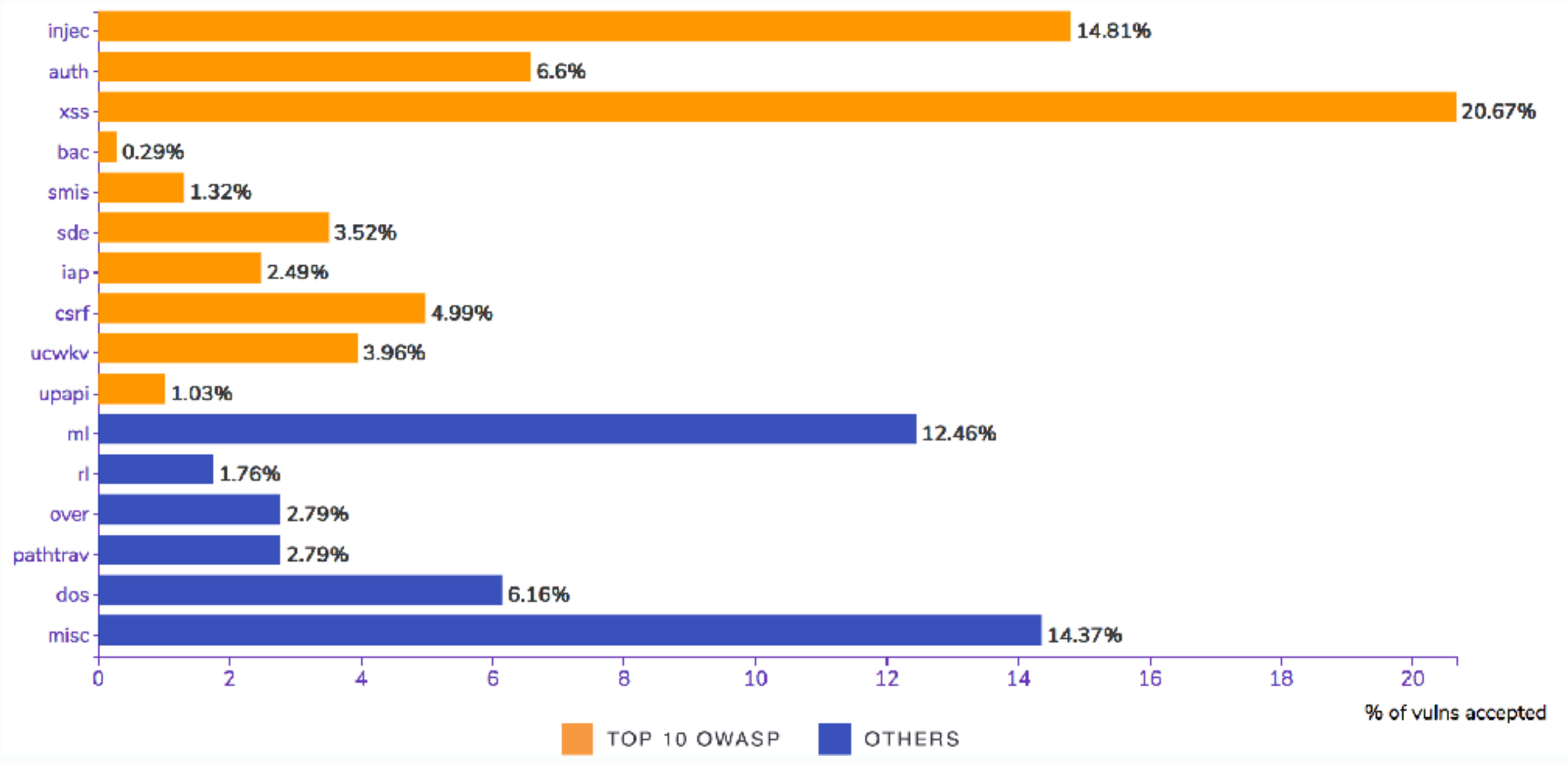
DATABASE OF REAL SECURITY VULNERABILITIES



*Accepted vulnerabilities per **year***

*Accepted vulnerabilities per **language***




injec
14.81%
auth
6.6%
xss
20.67%
bac
0.29%
smis
1.32%
sde
3.52%
iap
2.49%
csrf
4.99%
ucwkv
3.96%
upapi
1.03%
ml
12.46%
rl
1.76%
over
2.79%
pathtrav
2.79%
dos
6.16%
misc
14.37%
0
2
4
6
8
10
12
14
16
18
20
% of vulns accepted
TOP 10 OWASP
OTHERS

ANSWERING RESEARCH QUESTIONS

RQ1.1: *Is there **enough information available on OSS repositories** to create a **benchmark of software security vulnerabilities**?*

> There are enough vulnerabilities available on open-source repositories to create a database of real security vulnerabilities.

RQ1.2: *What are the **most prevalent security patterns on OSS** repositories?*

> The most prevalent security patterns are Injection, Cross-Site Scripting and Memory Leaks.

MAIN CONCLUSIONS

- We were able to retrieve vulnerabilities with an existence ratio of, approximately, **2.75** (682/248).

61M x **2.75** = **168 MILLIONS** of real security vulnerabilities

*Approximately, **246 thousand times larger than the current database.***

- It is possible to get a **considerable amount of vulnerabilities** identified by **CVE**.

- There is enough information on open-source repositories to create a database of real security vulnerabilities for different languages and patterns.

FUTURE WORK

- **Augment** the amount of security **vulnerabilities**, **patterns** and **languages** support.

- Continue **studying** and **collecting patterns** from GitHub repositories and extend to **other source code hosting websites** (e.g., bitbucket, svn, etc).

- Use **natural processing languages** to improve the mining tool.

THANK YOU!

Any Question?

GITHUB: WHY?

- More than **61M** of **repositories**, **22M users** and **199M issues**
- **Fast growth**
- Companies like **Microsoft**, **Google** and **Facebook**

GITHUB: WHY?

- More than **61M** of **repositories**, **22M users** and **199M issues**
- **Fast growth**
- Companies like **Microsoft, Google** and **Facebook**
- Top **10 of the trendiest programming languages**

- JavaScript - Python - PHP - C - C#
- Java - Ruby - CSS - C++ - Objective-C

GITHUB: WHY?

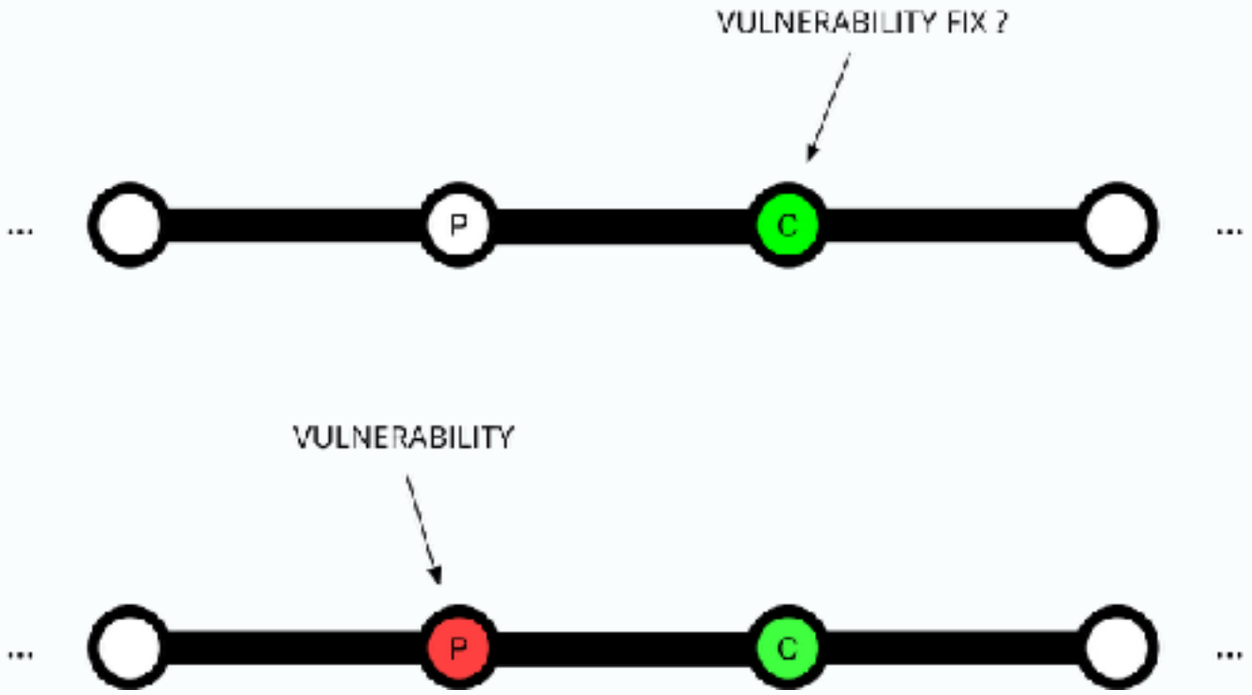
- More than **61M** of **repositories**, **22M users** and **199M issues**
- **Fast growth**
- Companies like **Microsoft, Google** and **Facebook**
- Top **10 of the trendiest programming languages**

- JavaScript - Python - PHP - C - C#
- Java - Ruby - CSS - C++ - Objective-C

HOW WERE THE REPOSITORIES CHOSEN?

- A repository must contain **one of the top programming languages on Github**;
- A repository needs to have a **size** of at least **2 commits**;

MINING TOOL



- Regular Expressions
- Syntactic Analysis on commits' messages

Limitation

Should we drop merges? ***NO.***

```
4 parents cc7e35b + f792943 + 0b0cda4 + 7b821a6    commit 082f6968bb204d1a3d8b2da3c53d6b7a59bbd985
```

WHY IS C ONLY THE SECOND?

- The type of mined patterns, since more than 50% of the patterns target web applications;

- The time when the repositories were migrated or created on Github (software migrated after years of development with lots of C files but small number of commits);

- low number of active repositories on Github compared with languages like JavaScript or Ruby (GitHut);

- Vulnerabilities from low level languages are more difficult to identify;

DATA VALIDATION

- Consistency checks on fields where the values were previously defined (e.g, vuln?, code, etc).

- Parent validation

- File existence check: some test cases were missing files due to the incorrect handling of exceptions in the early stages.

- A few times we corrected the missing packages on the cloud.

- Every 2 weeks, we ran a script in order to clean the non-viable samples form the cloud.

- Cardinality checks: Scripts to check if the information is in the right number or if there is any garbage ruining the data (e.g., if the total number of accepted vulnerabilities is equal to the sum of vulnerabilities for each pattern on the database)