

# Which Programming Languages Do Hackers Use?

## *Analyzing the Exploit Database with Python*

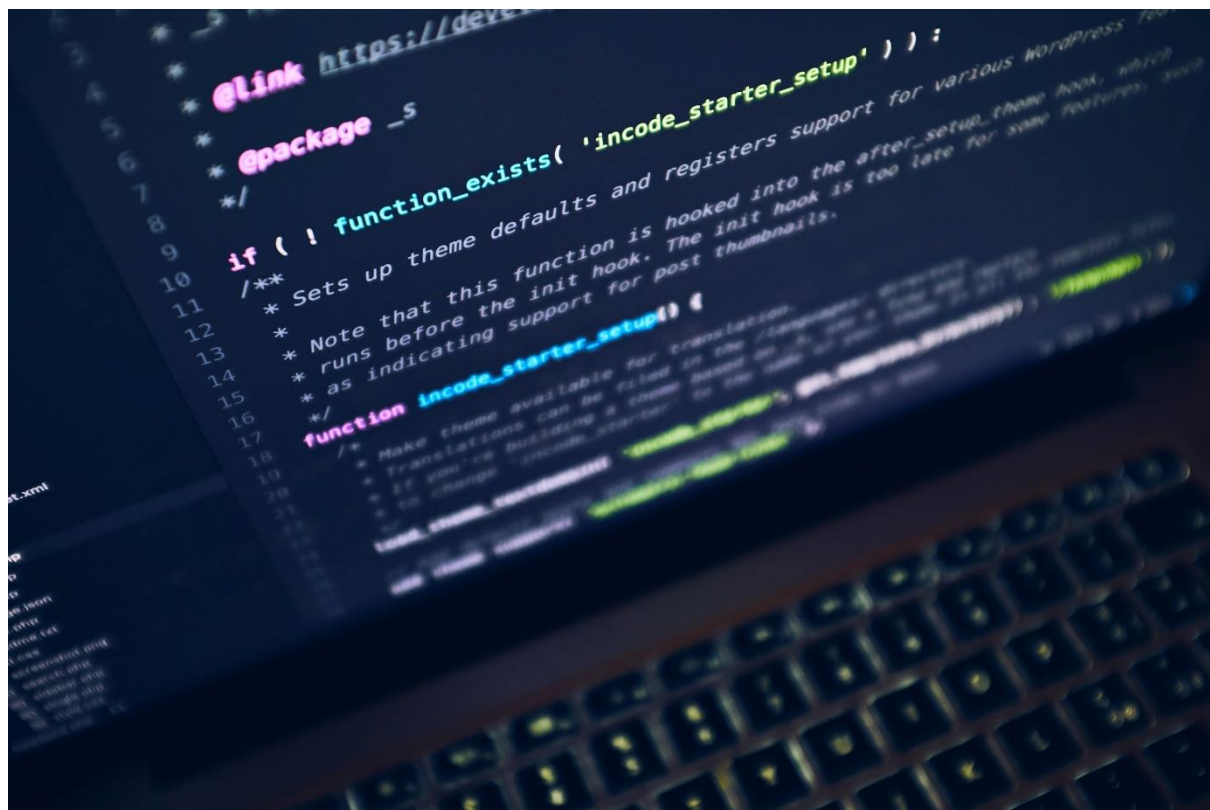


Photo by [Luca Bravo](#) on [Unsplash](#)

In 2021, together with fellow scientists, we conducted a [survey](#) at the German Chaos Computer Club (CCC). Our goal was to find out which programming languages are most commonly used by hackers. This article follows up the survey and compares its findings with an analysis of the [Exploit Database](#). Readers get step-by-step instructions on how to set up the analysis environment and a summary of the results. Thus, the article not only covers applied techniques, but also offers insights into the world of cybersecurity.

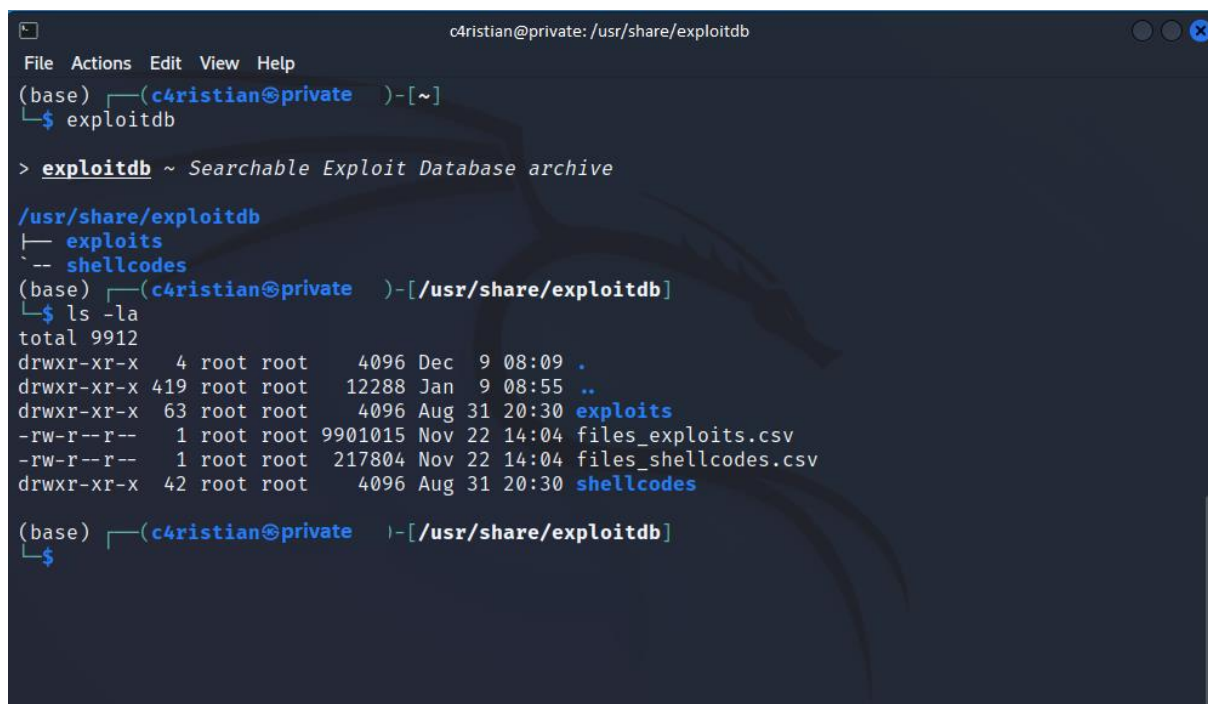
## Survey Overview

As Europe's largest association of hackers, the [Chaos Computer Club](#) provided a good basis for a study of programming languages prevalent in this field. We sent a link to an online questionnaire to the club's members in May 2021 for this purpose. According to the results, respondents mainly used the Shell and Python for hacking. Another key finding of the survey was that their language preferences had changed over time. In general, participants did not consider the choice of programming language to be essential for hacking. With 48 responses the sample size of the study was relatively small. As it only targeted members of the CCC, the findings might also be biased. Goal of this article is to validate the key results by analyzing the Exploit-DB.

# About the Exploit-DB

In the same year that the CCC survey was conducted, a so-called zero-day exploit attracted a lot of attention. Back then, the popular logging framework Log4j suffered from a highly critical [vulnerability](#). A script taking advantage of such a security flaw is called an *exploit*. As the name indicates, the Exploit Database offers a public archive of exploits and corresponding vulnerable software. Target audience of the platform are penetration testers and security researchers. Users can access the database via its website or a toolset available under [Kali Linux](#).

At the time of this writing over 45,000 exploits from more than 9,000 unique authors have been published in the Exploit-DB ([source](#)). Database entries consist of different attributes such as an exploit ID, author, type, and publication date. Each record relates to a file that contains the actual script or program of the exploit. Under Kali, we find the root directory of the database by executing the shell command `exploitdb`. Among other things, the directory contains a CSV file with an Exploit-DB snapshot, (`files_exploits.csv`), as well as sub directories with the actual scripts or programs (`exploits/`).



```
c4ristian@private: /usr/share/exploitdb
File Actions Edit View Help
(base) └─(c4ristian@private) -[~]
└─$ exploitdb

> exploitdb ~ Searchable Exploit Database archive

/usr/share/exploitdb
├─ exploits
└─ shellcodes
(base) └─(c4ristian@private) -[/usr/share/exploitdb]
└─$ ls -la
total 9912
drwxr-xr-x  4 root root   4096 Dec  9 08:09 .
drwxr-xr-x 419 root root 12288 Jan  9 08:55 ..
drwxr-xr-x  63 root root   4096 Aug 31 20:30 exploits
-rw-r--r--  1 root root 9901015 Nov 22 14:04 files_exploits.csv
-rw-r--r--  1 root root 217804 Nov 22 14:04 files_shellcodes.csv
drwxr-xr-x  42 root root   4096 Aug 31 20:30 shellcodes

(base) └─(c4ristian@private) -[/usr/share/exploitdb]
└─$
```

Figure 1: Directory of the Exploit-DB under Kali Linux

The comparison described in this article was prepared on 13 January 2023 using a database snapshot of 22 November 2022. Both the snapshot as well as the source code of the analysis are available on [GitHub](#).

## Setup and data transformation

In order to set up the analysis environment, the first step is to clone the GitHub project. All files required are stored in its root directory `exploits`. The implementation is based on the [Anaconda](#) Python distribution, which has to be preinstalled on the client machine. One can create and activate the conda environment by executing the following commands in the project's root directory:

```
conda env create -f environment.yml
conda activate exploits
```

The database snapshot to be analyzed is stored as a CSV file in the *data* folder. It was retrieved from Kali Linux and transformed with the script [execute\\_transformer](#). Both the time of transformation and that of the snapshot are documented in the file [timestamps](#). If needed, the snapshot can be updated by running the following commands in the Kali shell:

```
cp -p /usr/share/exploitdb/files_exploits.csv data/
python execute_transformer.py
```

The transformation script provides functions to tidy the data and to derive additional fields. One main task is to extract information on the used programming languages. To this end, the library [Pygments](#) is applied. Although its main purpose is syntax highlighting, the framework offers functions for guessing the programming language of a specific file.

Aside from Pygments, there are alternative language detection libraries. One example tested for this article is the deep learning solution [Guesslang](#). However, integrating it into the conda environment proved difficult and the processing time by far exceeded that of Pygments. Since Guesslang did not yield superior results, the latter framework was adopted. The following function includes Pygments into the data transformation script:

```
import pygments
from pygments.lexers import guess_lexer_for_filename

def _parse_exploit_file(file_name):
    with open(file_name, encoding="UTF-8") as file:
        lines = file.readlines()
        text = "\n".join(lines)

        line_count = len(lines)

        try:
            lang_guessed = guess_lexer_for_filename(file_name, text).name
        except pygments.util.ClassNotFound:
            lang_guessed = None

    return line_count, lang_guessed
```

The above Python code reads a certain file, counts its lines and uses the function `guess_lexer_for_filename` to detect the programming language. To achieve this, the framework applies various Lexers, meaning classes for syntax analysis. Only those Lexers assigned to the given file extension are considered. There are suffixes for which only one class exists, with others the choice is ambiguous. For example, the extension “py” is clearly assigned to Python, while the suffix “pl” could either point to Perl or Prolog. The best matching Lexer is returned as the result. Its name reveals the programming language and builds the basis of the analysis. The findings discussed in the next section originate from the notebook [comparison](#). In addition, the GitHub project provides further notebooks to explore specific aspects of the Exploit-DB.

## Discussion of results

Central question of the CCC survey was, which programming languages participants had used in the year prior to the study. Respondents could choose more than one answer option. Figure 2 compares the top ten languages mentioned by CCC members with those used by Exploit-DB authors. The chart reveals similarities, but also differences.

First, there is a major gap in sample sizes. Overall, 48 members of the CCC took part in the survey. In contrast more than 900 unique authors published over 2,500 files in the Exploit-DB in the years 2020/21. These two years were selected as they coincide with the research period of the comparison study. In order to avoid duplicates, each author-language combination in the Exploit-DB was counted only once. For the research period, this led to 1,134 language references, of which 1,116 were part of the top ten. On the other hand, participants of the CCC survey named their top ten languages 140 times.

CCC & Exploit-DB: Top 10 languages 2020/21

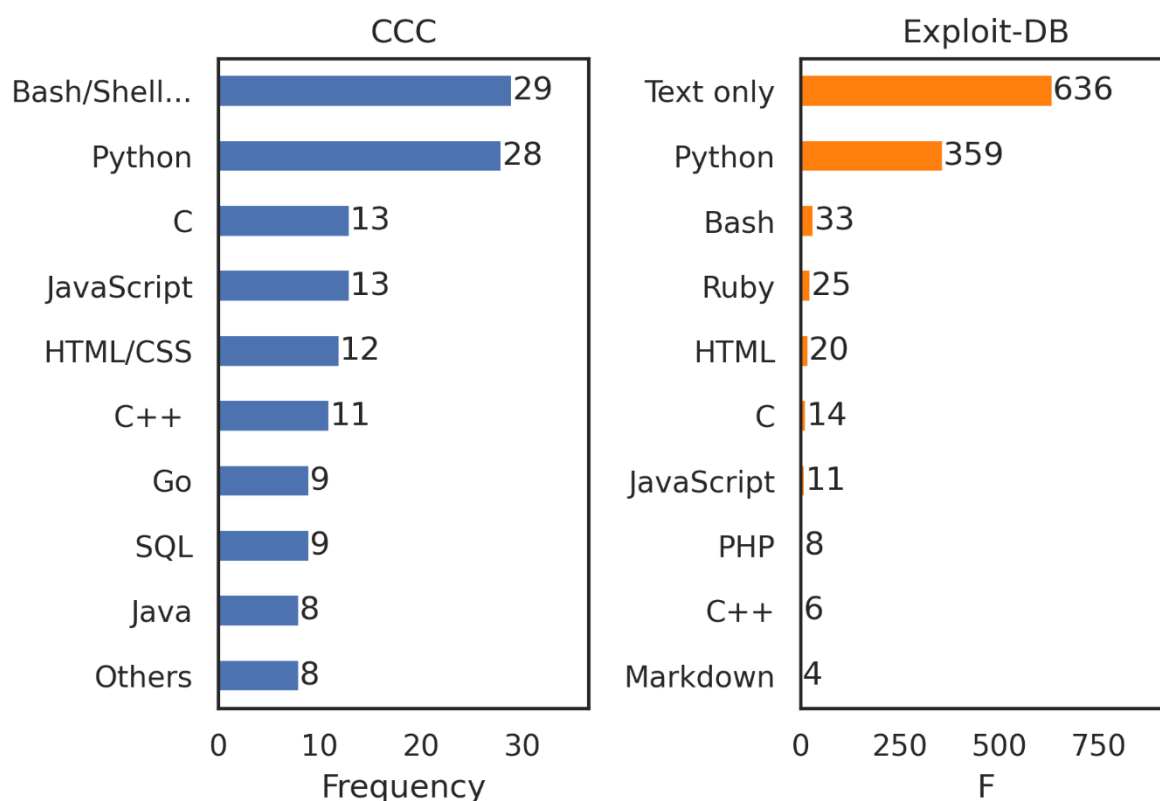


Figure 2: Top ten languages of the CCC survey and the Exploit-DB

Members of the CCC mainly used the Shell (Bash/Shell/PowerShell) and Python, followed by C, JavaScript, and HTML/CSS. All of these technologies appear as well in the Exploit-DB. Generally, there is a substantial overlap in language choices. Six out of ten technologies are present in both lists in Figure 2. Python consistently ranks second, showing its popularity in the cybersecurity domain. Yet some languages on one side of the chart do not appear on the other.

One main difference is the unbalanced distribution of languages in the Exploit-DB. More than half of the submissions were text files. Here, the name “Text only” assigned by Pygments is

misleading. Text files in the Exploit-DB typically include a description, but also often shell commands and possibly scripts in other languages. Therefore, certain technologies might be underrepresented in the results. Spot checks have shown that this is likely to be true for shell scripts, ranking first in the CCC study. Herein lies a limitation of our approach. Frameworks like Pygments have problems in assessing multi-language files. Overcoming this issue would be an interesting topic for a follow-up study.

Let us move away from the years 2020/21 and have a look at the entire Exploit-DB history. Figure 3 shows the top ten languages of all time in the database. As above, language detection relied on Pygments and each author-language combination was counted only once.

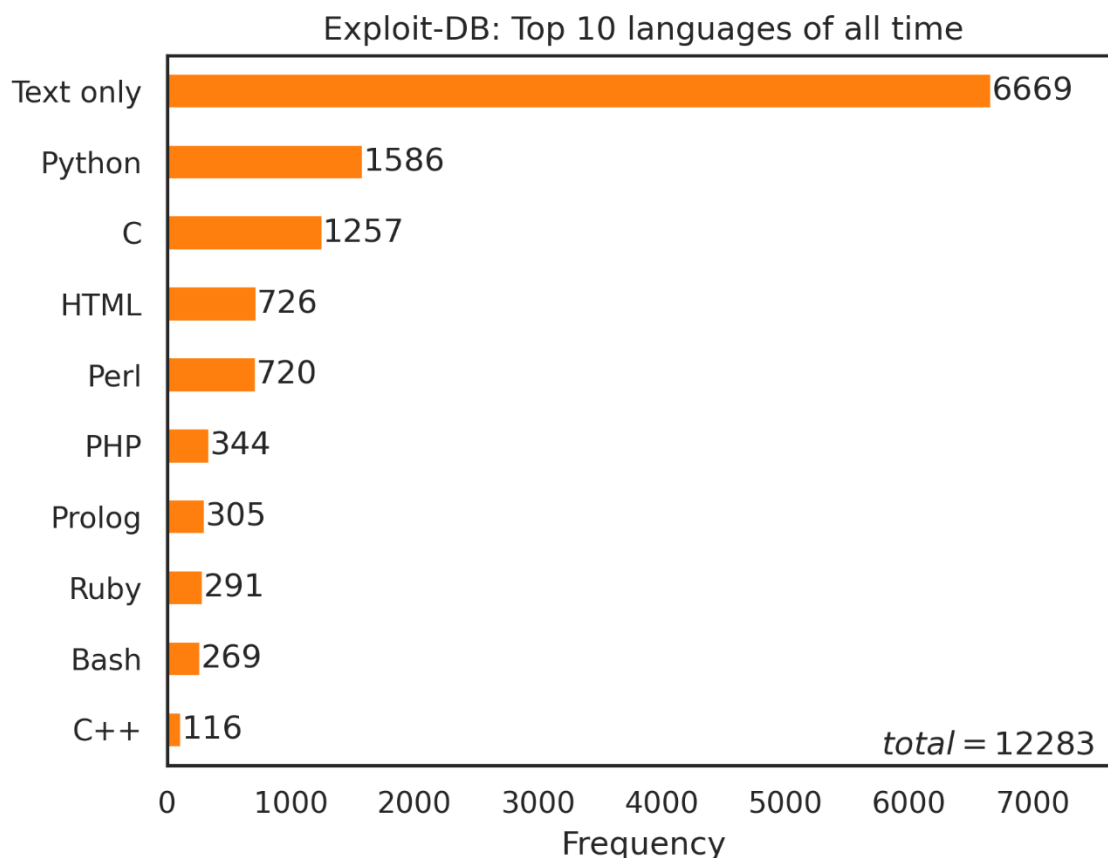


Figure 3: Top ten languages in the Exploit-DB of all time

According to Figure 3, the top ten languages of all time in the Exploit-DB are led by Text, followed by Python, C, HTML, and Perl. Again, there is a considerable intersection with the choices of CCC members. Certainly a surprise is the presence of Prolog in the second half of the list. Presumable reason for this is a wrong classification of files with the extension “pl”. In any case, the relatively high rank of Perl is remarkable as the language does not appear at all in Figure 2. This coincides with another finding of the CCC survey. In the study, a majority of participants (77.5%) reported that their language preferences had changed over time. To evaluate this, we can take a look at Figure 4. The chart visualizes the percentage share of the top ten languages in the Exploit-DB over 25 years prior to this analysis. Each author-language combination was counted once per year. As a result, we found 16,422 language references from 9,592 unique authors in the period under study.

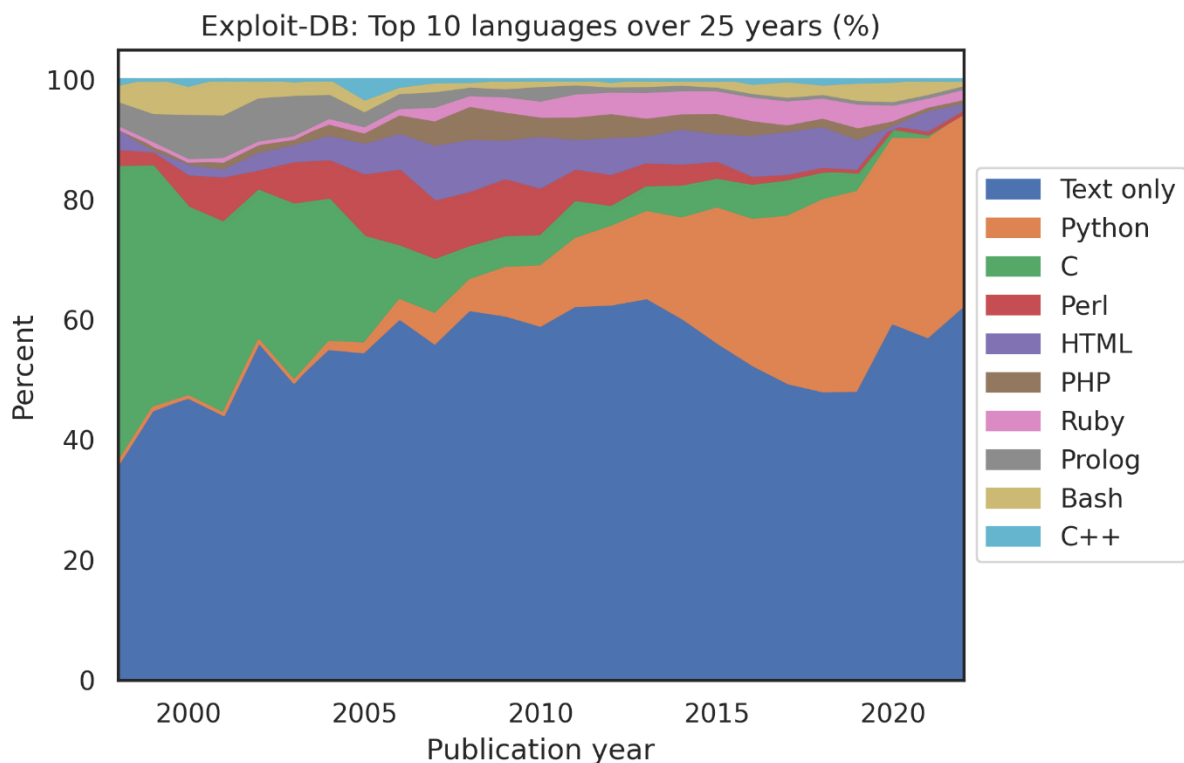


Figure 4: Top 10 languages of the Exploit-DB over 25 years preceding the analysis

Figure 4 reveals that the percentage of text files in the Exploit-DB remained rather stable. The time frame of 25 years was chosen because the number of submissions prior to this period was relatively low. Aside from a constant use of Text, Figure 4 shows a change in preferences for the other languages. Most obvious is a shift from C towards Python in the database. One possible reason is mentioned in the CCC study as well. Participants reported that they did not consider the choice of programming language to be vital for hacking. Thus, the increasing use of Python could simply reflect its general rise in popularity in recent years. Language preferences should therefore continue to change in future as technology evolves. Such a shift could be triggered, for example, by the introduction of [Rust](#) as a second language for the development of the Linux kernel. Whether this forecast proves true will be seen in the coming years.

## Conclusion

In summary, the comparison presented in this article reveals a substantial overlap between languages used by CCC members and Exploit-DB authors. Both datasets confirm the popularity of Python in the cybersecurity domain. Moreover, each dataset indicates a change in language preferences over time. One possible reason is formulated by the CCC study. Participants did not consider the choice of programming language to be essential for hacking. Following this explanation, one should expect language preferences to continue to change with technological progress. One major limitation of the analysis of the Exploit-DB relates to the language detection approach. Due to problems with multi-language files, certain technologies are possibly underrepresented. Solving this issue would be an interesting topic for a follow-up study. Clearly, the Exploit-DB offers a rich dataset for both data scientists and security experts. There is much more to learn about the art of exploitation.

All images unless otherwise noted are by the author.

## Further Reading

C. Koch, K. Müller and E. Sultanow, *Which programming languages do hackers use? A survey at the German Chaos Computer Club*, <https://arxiv.org/abs/2203.12466>, arXiv, 2022.