Improving Password Generation Through the Design of a Password Composition Policy Description Language

Anuj Gautam, Shan Lalani, and Scott Ruoti, *The University of Tennessee*

https://www.usenix.org/conference/soups2022/presentation/gautam

This paper is included in the Proceedings of the Eighteenth Symposium on Usable Privacy and Security (SOUPS 2022).

August 8–9, 2022 • Boston, MA, USA

978-1-939133-30-4

Improving Password Generation Through the Design of a Password Composition Policy Description Language

Anuj Gautam
The University of Tennessee

Shan Lalani
The University of Tennessee

Scott Ruoti
The University of Tennessee

Abstract

Password managers help users more effectively manage their passwords, yet the adoption of password generation is minimal. One explanation for this problem is that websites' password composition policies (PCPs) can reject generated passwords, creating a usability impediment. To address this issue, we design a PCP language that websites use to describe their PCP and that managers use to generate compliant passwords. We develop this language using an iterative process involving an extensive collection of PCPs scraped from the Web. We provide libraries for adopting our PCP language into websites and password managers and build proof-of-concept prototypes to verify the real-world feasibility of our PCP language. Using a 25-person user study, we demonstrate that our language and libraries are easy to pick up and correctly use for novice developers. Finally, we replicate and extend past research evaluating Web PCPs, showing that half of PCPs fail to require passwords that resist offline attacks when considering that users prefer certain character classes when selecting their passwords.

1 Introduction

Despite their problems [7–9, 27, 30, 34, 37], passwords remains the dominant form of authentication [5]. Password managers strengthen password-based authentication by helping users generate, store, and enter passwords, making it easier to adopt strong, unique passwords [19, 27]. Still, research has shown that password manager users underutilize password generation [19, 28]. One potential explanation for this phenomenon is that websites' password composition policies (PCPs) can reject generated passwords, decreasing the usability and utility of the generator. [16, 25].

To address this issue, we design a PCP language that websites can use to encode and publish their PCP, with password managers downloading the PCP to ensure that they only generate compliant passwords. To inform the design of this PCP language, we extract 270 PCPs from a geographically diverse set of 626 popular websites. Using this dataset, we build an initial PCP language, then iteratively refine it as we encode the gathered PCPs, stopping once all PCPs in our data set can be efficiently and useably encoded. Our final PCP language is more feature-rich than previous efforts and is the first PCP language that can represent the full range of PCPs found in our dataset.

To demonstrate the feasibility of our proposed language, we (i) build proof-of-concept websites that publish their PCP using our language; (ii) modify BitWarden, a popular password manager, to download these PCPs and generate compliant passwords; and (iii) create Python and JavaScript libraries that make it easy to use our PCP language in server- and client-side code. Next, we conduct an online usability study with 25 participants, measuring their ability to author PCPs using our language and tools. Our results show that most participants can rapidly comprehend our language and author PCP descriptions, even for complex policies.

Finally, we replicate and extend prior work analyzing Web PCPs [10, 20]. In contrast to prior efforts that use a simple heuristic that only considers the minimum length and allowed characters for measuring PCP strength, our analysis takes into account all requirements of the PCP. Additionally, our analysis includes both upper- and lower-bound estimates for PCP strength that take into account how users select passwords [18, 36]. This improved analysis shows that most PCPs in our dataset fail to require passwords that resist offline attacks. Furthermore, for users that prefer passwords comprised primarily of digits [18], nearly half of the evaluated PCPs fail to require passwords that resist online attacks.

USENIX Symposium on Usable Privacy and Security (SOUPS) 2022.
August 7–9, 2022, Boston, MA, United States.

Research Artifacts: Our data, scripts, and prototypes are available at https://userlab.utk.edu/publications/gautam2022improving.

2 PCP Dataset

To inform the design of our PCP language, we gathered an extensive corpus of PCPs deployed on the Web. Our sample is demographically diverse, including websites from highly-populated countries in each of the six inhabited continents: Africa—Nigeria, Asia—India, Europe—Germany and the United Kingdom (UK), Oceania—Australia, North America—United States (US), South American—Brazil. We also measured PCPs from China, Iran, and Russia to see if their high levels of Internet censorship [15] impacted PCP selection.

2.1 Sources

We used the Alexa and Quantcast lists of the most popular websites to select websites for each country. In January 2019, we downloaded the Alexa lists of the 250 most popular US websites and the top 50 lists for the remaining nine countries we examined. As we began to analyze these websites, we noticed a high overlap between the websites listed for each country. To obtain more unique websites for each country, in February 2019, we downloaded the Quantcast lists of the top 50 most popular websites for each country. We selected Quantcast as its country-specific lists had minimal overlap with global and US-specific websites from Alexa. We also analyzed the websites listed in the Quantcast top 50 global lists. In total, these lists identified 626 unique websites.

Next, we removed websites that do not support account creation, delegate all authentication to single sign-on (SSO) providers, or require resources we do not have to create an account (e.g., a bank account). For the remaining 320 websites, we identify websites that use the same authentication backend (e.g., google.com and youtube.com), keeping only a single representative website. We then extracted PCPs from the remaining 270 websites.

2.2 Analysis

To extract the PCP for each website, we took the following steps. First, we would look for PCP components described textually on the account creation web page or elsewhere on the domain. Second, we would examine the HTML form, looking for validation attributes that restricted what users could enter for their password. Third, we evaluated any JavaScript used to validate the password, identifying restrictions enforced therein. Fourth and finally, we manually tried to enter various passwords of different lengths and compositions.

2.3 Limitations

While our data collection resulted in a large and rich corpus, we recognize there are limitations to our methodology. First, while covering more features than past efforts [10, 14, 20], our data is not comprehensive. Still, we believe our dataset is sufficient for our purposes as we achieved saturation [2]—i.e., we stopped discovering new PCP features at the latter end of our analysis.

Second, it is likely that we missed some PCP edge cases. Only by investigating the server-side code would it be possible to identify the exact PCP definitively. Automating the process to check more password combinations would be problematic as this would involve flooding the website with passwords.

3 PCP Description Language

Using our PCP dataset, we design a language for describing PCPs. Our language has two key design goals: (1) describe the PCPs in our dataset and (2) be simple to read and write for administrators and machines. To achieve these goals, we followed an iterative design process:

First, we created a draft version of our PCP language based on prior research (§9) and PCP features in our dataset. Second, we encode the PCPs in our data set using this language. When we encountered a PCP that was onerous to encode, we modified our draft PCP language to address pain points. We would then re-encode all prior PCPs to ensure that our change did not cause a usability regression. Third, after encoding all PCPs, we reviewed our language with others from our research group, focusing on improving the language's readability and identifying PCP features they had encountered in the wild but are absent in our PCP dataset. Based on their feedback, we updated our language and re-encoded the PCPs in our dataset (continuing to look for usability issues). After making a full pass encoding PCPs without changing our language, we considered it finished.

3.1 PCP Language

A PCP in our language is composed of two components: (a) a set of characters allowed in a password and (b) rules about password composition.

The allowed characters are grouped into named, disjoint sets of characters—a *charset*. By default, the PCP uses the following four default charsets: lowercase English letters (*lower*), uppercase English letters (*upper*), Arabic numerals (*digits*), and the OWASP password symbols [26] (*symbols*). Our language allows these default charsets to be modified, new charsets to be added, and default ones to be removed. Our language also provides an *alphabet* charset that, if used, merges and replaces the default *lower* and *upper* charsets.

A PCP composition rule is a set of *requirements* that passwords must comply with to be valid. If a PCP contains multiple rules, a password need only satisfy the requirements for a single rule to be valid (the overwhelming majority of PCPs only have one rule). For example, if one rule specified that passwords must be eight characters long and contain lowercase letters and symbols and another rule specified that passwords must be fifteen characters long, fifteen character passwords of only digits would be valid, whereas fourteen character passwords of only digits would not.

The possible requirements in each rule are as follows:

- *min_length* is a positive integer specifying the password's minimum length (inclusive). All rules require that `min_length` is set, with all other requirements optional.
- *max_length* is a positive integer specifying the password's maximum length (inclusive).
- *max_consecutive* is a positive integer indicating the maximum number of times the same character can appear consecutively in a password. For example, to prevent passwords such as *AAA* or *ZZZ*, *max_consecutive* would be set to 2.
- *prohibitted_substrings* is a set of strings that may not appear anywhere in the password. When used, this commonly includes the website name and other related words. For example, to prohibit the string "google", *prohibited_substrings* would be set to *["google"]*.
- *require* is a list of charsets that must appear in the password. For example, to require that a password must have letters and digits, *require* would be set to *["alphabet", "digits"]*.
- *require_subset* is an object containing a list of charsets (*options*) from which *count* of those options must appear in the password. For example, to require that a password must have digits and symbols, but not necessarily both, *require_subset* would be set to *{"options": ["digits", "symbols"], "count": 1}*. If not set, *options* defaults to using all the PCP's charsets; *count* defaults to one.
- *charset_requirements* is a map between charset names and requirements for the named charset. For example, to add additional requirements for digits, *charset_requirements* would be set as such: *{"digits": {requirements}}*. Possible requirements include:
  - *min_required* is a positive integer specifying the minimum number of times this charset must appear in the password.
  - *max_allowed* is a positive integer specifying the maximum number of times this charset may appear in the password. For example, if set to two for the digits charset, passwords containing *111* or *123* would be rejected.
  - *max_consecutive* is a positive integer indicating the maximum number of times this charset can appear consecutively in a password. For example, if set to two for the alphabet charset, passwords containing *abc* or *ddd* would be rejected.
  - *required_locations* is a list of indices for the password at which this charset must appear. Passwords are zero-indexed and negative indices are supported (i.e., reverse string indexing). For example, to require a password that starts and ends with a symbol, *required_locations* for the symbols charset would be set to *[0, -1]*.
  - *prohibited_locations* is a list of indices for the password at which this charset must *not* appear. Passwords are zero-indexed and negative indices are supported (i.e., reverse string indexing). For example, to prevent a password from having the last two characters as digits, *prohibited_locations* for the digits charset would be set to *[-1, -2]*.

```
{
  "charsets": {
    "name": "characters", ...
  },
  "rules": [{
    "min_length": ℤ+,
    "max_length": ℤ+,
    "max_consecutive": ℤ+,
    "prohibited_substrings": ["substring", ...],

    "required": ["charset_name", ...],
    "require_subset": {
      "options": ["charset_name", ...],
      "count": ℤ+
    },

    "charset_requirements": {
      "charset_name": {
        "min_required": ℤ+,
        "max_allowed": ℤ+,
        "max_consecutive": ℤ+,
        "required_locations": [ℤ+, ...],
        "prohibited_locations": [ℤ+, ...],
      }, ...
    }
  }, ...]
}
```

Listing 1: JSON schema for our PCP language

A JSON schema for our final PCP language is given in Listing 1. Examples of real-world PCPs encoded using our language are given in Listing 2.

Examining the JSON-encoded PCPs in our dataset, we find that they are 17–205 characters long, with a median length of 36 characters. These small sizes are evidence that our PCP efficiently encodes passwords. Lastly, we note that while we used JSON to encode policies, they could also easily be encoded in a wide range of data-interchange formats (e.g., YAML, protobuf).

```
# Passwords of length six to twleve (walmart.com)
{"min_length": 6, "max_length": 12}

# Password must include at least one digit, symbol, and
    alphabetic character (facebook.com)
{
  "min_length": 6,
  "require": ["digits", "alphabet", "symbols"]
}

# Custom definition for symbols that are allowed
    (macys.com)
{
  "charsets": {"symbols": "!\"#\$%&'()*+:;<>?@[]^`{}~"},
  "rules": [{"min_length": 7, "max_length": 16}]
}

# Password must have at least one alphabetic character
    and either a digit or a symbol (bbc.com)
{
  "min_length": 8,
  "max_length": 50,
  "require": ["alphabet"],
  "require_subset": {
    "count": 1,
    "options": ["digits", "symbols"]
  }
}

# Password can be eight characters if it contains a
    lowercase character and a digit. Otherwise, it must
    be fifteen characters long. (github.com)
{
  "rules": [
    {"min_length": 8, "require": ["lower", "digits"]},
    {"min_length": 15}
  ]
}
```

Listing 2: PCP examples encoded in our language

4 PCP-Compliant Password Generation

To demonstrate the feasibility of our proposed language, we (1) created libraries for using our PCP language, (2) built proof-of-concept websites that publish their PCP using our language, and (3) modified a password manager to generate PCP-compliant passwords.

4.1 Library Implementations

We constructed Python[1] and JavaScript[2] libraries to support our PCP language. These libraries enable the programmatic creation of PCPs, encoding PCPs to JSON, and parsing PCPs from JSON. They also automatically validate PCPs to ensure they are both semantically correct—e.g., that *min_length* is appropriately set and that character sets do not overlap—and logically consistent—e.g., that a policy does not simultaneously require and prohibit a character class.

These libraries also support checking passwords against a PCP. Finally, they can evaluate the strength PCPs, giving administrators an idea of how likely a PCP is to result in passwords that resist online and offline guessing attacks (see Appendix B for more details).

4.2 Website Implementation

We built five proof-of-concept websites, each with a PCP of varying complexity. We implemented these websites using Flask (Python) on the backend and JavaScript on the frontend. Each website publishes its PCP and provides a form where passwords can be generated, submitted, and verified.

We identified three approaches for publishing PCPs:

1. **HTML:** A new attribute could be added to the password field, which would be set to the JSON-encoded PCP. Alternatively, the PCP could be encoded as XML within the HTML, adjacent to the password field.
2. **HTTP header:** An HTTP header (e.g., *X-PCP*) can specify the JSON-encoded PCP for relevant pages.
3. **File:** The JSON-encoded PCP could be available at a known URL (e.g., `domain.tld/pcp.json`). If there are multiple PCPs for a domain, this file could contain a mapping between URLs and PCPs.

Our websites use the third approach as it is the easiest to implement and the only approach which can work with non-browser-integrated managers. We checked the validity of submitted passwords on the client-side using our JavaScript library and on the server-side using our Python library. *A significant benefit of publishing PCP and using our tool to validate them is that if the PCP is ever updated, there is no need to separately update the validation code, simplifying developer workloads and preventing situations where the client- and sever-side validation may become out of sync.*

4.3 Password Manager Implementation

We modified BitWarden, a popular open-source password manager, to check if a domain hosts a `/pcp.json` file, and if so, to use it to generate PCP-compliant passwords. The actual generation is handled by our JavaScript library and occurs over three phases:

In the first phase, we set the password length to the smallest *min_length* (if there are multiple rules). Next, we use our JavaScript library to check if passwords of this length using this PCP will be offline-resistant password [11]. If not, we choose the smallest length that would result in an offline-resistant password.

In the second phase, we create an array of length equal to our calculated minimum length. Each position within the array contains an (initially empty) list of which charsets can appear at that position. To fill these lists, we first satisfy *required_locations* by setting the list at the specified index to its respective charset. Next, we set the remaining empty lists as necessary to satisfy *min_required* and *required*. Lastly, the

[1] https://pypi.org/project/password-policy/

[2] https://www.npmjs.com/package/password-composition-policy

remaining empty lists are set to include all allowed character sets unless doing so would violate *max_allowed*.

In the third phase, we shuffle all indices not set due to *required_locations*. We then generate a password by randomly selecting a character at each index from the charsets in the list at that index. We then check the generated password against the other requirements in the PCP. If it is not, we repeat phase three until we generate a valid password. In addition to ensuring that generated passwords are PCP-compliant, we also follow recommendations by Oesch et al. [24] and ensure that generated passwords are not randomly weak. This is done by checking passwords using zxcvbn and ensuring that the generated passwords receive the highest strength rating (4).

5 Usability Study

To evaluate the usability of our developed language and libraries, we conducted an IRB-approved user study wherein participants authored five PCPs of varying complexity using our PCP language. This section gives an overview of the study and describes the tasks and study questionnaire. In addition, we discuss the development and limitations of the study. The study instrument is given in Appendix A.

5.1 Study setup

The study ran for three weeks starting Friday, January 28, 2022, and ending Tuesday, February 15, 2022. In total, 25 participants completed the study. The study was designed to take about thirty to forty minutes and participants were compensated with a $25 Amazon gift card. Participants were required to have Python 3.6.1 or higher installed on their system. The study was administered online using Qualtrics.

Participants were recruited from the EECS department at our local university using posters, email invitations, and class announcements. We also asked researchers at other universities to share the study with their students. We chose to use EECS students as we felt they were a good representation of novice developers, and we hypothesized that our language and libraries would be sufficiently usable to support novice developers.

5.2 Study tasks

Participants started by reading and accepting an informed consent statement. Next, participants installed our Python library and executed a Python instruction that allowed us to confirm that the library was correctly installed. They then entered basic demographic information (class standing, major, gender).

Participants were told that in the study they would be authoring five PCPs. They were given a link to documentation for the Python library and informed that this link would also be provided with each task. The documentation included a description of our language, source code examples, and JSON-encoded PCPs.

Participants encoded five PCPs:

1. The password must be at least 8 characters.
2. The password must be at least 8 characters and contain at least two of the following: uppercase, lowercase, digits, symbols.
3. The password must be at least 12 characters, contain a letter and a number, and not contain whitespace.
4. The password must be at at least 8 characters long and contain a letter and a number. Alternatively, the password must be at least 15 characters.
5. The password must be at least 8 characters, contain at least two symbols, contain either an upper or lowercase letter, not contain the string "mywebsite", and none of the following characters: `^'";/\`

Upon submitting a PCP, the survey checked whether the submitted PCP was parsed correctly. It also verified that the PCP was correct by checking two valid and two invalid passwords. Participants were allowed to continue when they submitted a correct PCP description or once two minutes had passed (to prevent participants from becoming stuck). After submitting their policy, participants completed an After-Scenario Questionnaire [31] (ASQ) about their experience.

Upon completing all five policies, participants were asked to fill out the System Usability Scale [6] (SUS) regarding their overall experience. They were also asked what they liked most and least about the system and library. Finally, they were asked to provide any other feedback they had.

5.3 Demographics

Participants were largely male: male (19; 76%), female (6, 24%). All students studied computer science (23; 92%) or electrical engineering (2; 8%). Participants were all more senior students: juniors (2; 8%), seniors (10, 40%), graduate students(13, 52%).

5.4 Study Design

Initially, we structured study compensation as a raffle, where five participants would receive a $50 Amazon gift card. Under this incentive scheme, only two participants completed our study. This led us to revise our study to compensate every participant (including the two who had already completed it). After making this revision, re-obtaining IRB approval, and re-launching the study, we quickly gathered our remaining 23 participants.

We also changed our documentation between the two iterations of our study. Initially, the survey provided a link to the documentation explaining how to author policies in JSON, with that documentation providing a link the Python

Policy	Correct	JSON mistakes	Minor errors	Major errors	Mean time in minutes	Mean ASQ
1	92%	0	0	2	1.5	7.0
2	92%	1	0	1	1.4	6.7
3	88%	1	2	1	4.6	5.7
4	96%	1	1	0	0.6	6.3
5	64%	3	7	0	4.0	6.0

Table 1: Quantitative results by policy

library's documentation. However, after looking at the first two participants' results, it became clear that they lacked proficiency in JSON. To encourage participants to use the Python library, we changed the survey's documentation link to point to the Python library's documentation, with that documentation providing a link to the JSON documentation. Participants could still directly author JSON, and eight (40%) did for at least one task.

5.5 Limitations

Our students do not have the same experience as the administrators responsible for authoring PCPs. Similarly, participants had less incentive to learn and correctly enter policies than administrators trying to use these tools. As such, our results may not fully represent the usability of our tooling for the target audience. However, past research has shown that students can serve as a reasonable approximation for developers [22, 23].Lastly, our study only measured the ability of participants to author policies, not to read them.

6 Study Results

In this section, we report the significant findings of our user study. Quantitative results for each policy are given in Table 1. Mean completion times use the geometric mean [31].

6.1 Success Rates

Overall, participants did very well at encoding policies. Two participants struggled at nearly all tasks, only correctly encoding a single PCP. Excluding them from our data, completion rates move to 100%, 100%, 96%, 100%, and 68%, respectively.

In policies, we detected three types of errors. First, incorrectly formatted JSON (6 total), likely stemming from unfamiliarity with JSON. Second, minor errors (10 total), such as forgetting to include a prohibited character or including a rule from a previous policy. We only classify errors as minor if users showed comprehension of the tested language and library features but made an error with the values used. Third, major errors (4 total) resulting in an entirely incorrect submission. These errors indicate that participants failed to understand how to use the language and library.

Looking at Policy 5's results more closely, we see that three errors (12%) arose due to incorrectly encoded JSON, with the remaining seven (28%) arising due to participants forgetting to include one or more of the prohibited characters. This happened even though these same participants had properly excluded characters in Policy 3.

6.2 Completion Times

Participants generally completed tasks quickly, with (geometric) mean times ranging between 36 seconds and 4 minutes. However, we note that these times are lower bounds as they do not include time participants may have spent reading documentation between tasks and before they started interacting with the task. Still, these times suggest that it is easy to pick up and use our language and library with no prior experience.

Using a two-way ANOVA, we find that while there is a statistically significant difference between how long each policy took to create ($F(4,170) = 8.731$, $p < 0.001$), though this is not surprising given the difference in difficulty between policies. We do not find a statistically significant difference between time taken to author PCPs using JSON or our library ($F(1,170) = 0.109$, $p = 0.74$), nor for the interaction effect ($F(4,170) = 0.027$, $p = 1.00$). This is a surprising result as, based on our first two respondents, we expected participants to struggle authoring JSON.

6.3 Perceived Usability

Overall, policies received good ASQ scores (see Table 1), indicating that it was easy and relatively quick to author policies. The mean SUS score was 65, which can be interpreted as "Good" usability [3], receives a C grade [31], and is just above the 40th percentile of systems studied with SUS. While this is an acceptable score for our language and library to be used in the wild [3], it still fell short of our initial expectations.

Looking into the qualitative feedback, we discovered three primary critiques of our tooling. First, many participants felt that JSON was confusing. Second, participants wanted additional documentation. While we provided one example for every PCP feature, they wanted even more. Third, participants were confused by our library providing two ways to create PCPs: (a) a class exactly matching the JSON schema and (b) a simplified class that could be used to encode simple PCPs more directly. While we created this second method to reduce the amount of code participants needed to write for simple PCPs, it ended up causing unneeded confusion and is a prime candidate to remove from our library.

6.4 Takeaways

Overall, our results show that our proposed language is promising, though it has room for improvement. Other than the two participants who failed all but one task, every other participant correctly encoded Policies 1–4, except for one mistake in Policy 3.

However, of these 23 participants, nine (39%) submitted incorrect solutions for Policy 5. One-third of these errors (3) arose from improperly encoded JSON. This suggests that in line with participant feedback, it might be worthwhile to consider other more developer-friendly encodings (e.g., YAML) or supporting multiple encodings, allowing developers to choose which they will use. Alternatively, pushing for programmatic specification of PCPs could be used to avoid encoding issues entirely.

Two-thirds of the errors (6) for Policy 5 arose from minor issues with the PCP. Half of these issues (3) involved the participants removing some but not all of the prohibited characters from the symbols list. This may have arisen as the textual policy described a denylist for restricted characters, whereas participants chose to create an allowlist of symbols. To address this, the library could allow users to specify a denylist for characters and then have the library generate the appropriate character set, though further research would be needed to measure the efficacy of this approach.

The other issues with Policy 5 (3) arose from participants failing to include the list of restricted characters, even though the other requirements for this policy were included. This happened even though these same participants had properly excluded characters in Policy 3. It is unclear whether this issue stems from something in the design of our language, the general challenge of remembering all the requirements in a complex policy, or study fatigue.

7 Website Analysis

Using the PCP dataset we collected to build our language, we replicated and extended prior work analyzing website PCPs [10, 20]. Our analysis covers (1) the strength of PCPs, (2) the requirements used in PCPs, and (3) additional non-PCP authentication-related details.

To estimate PCP strength, we calculate the average number of guesses an adversary would need to discover a password that (a) complies with the PCP and (b) is of the smallest allowed length. In contrast to previous work [10, 20] which calculates strength based only on the smallest allowed length and count of allowed characters (i.e., $\#characters^{length}$), our estimates take into account all PCP features. First, we create a canonical representation of the PCP. Second, we enumerate all unique password compositions—a password composition specifies the number of characters from each character class that makes up a password. Third, for each password composition, we calculate the number of unique passwords that exist for that composition, reducing this number to account for passwords that fail to meet the various *chartset_requirements*. Finally, we sum these counts. A more detailed description of this algorithm is given in Appendix B.1.

Country	Count
Global	65
Australia	13
Brazil	14
Germany	17
India	9
Nigeria	13
UK	8
US	72
China	28
Iran	12
Russia	19

Popularity	Count
Top 10	8
Top 50	24
Top 100	25
Top 500	59
Top 1000	25
Top 5000	79
5000+	50

Use case	Count
E-commerce	58
Finance	10
News	72
Social media	55
Software	13
Streaming	28
Other	34

Ad Provider	Count
Yes	158
No	112

Public username	Count
Yes	43
No	227

Past breach	Count
Yes	51
No	219

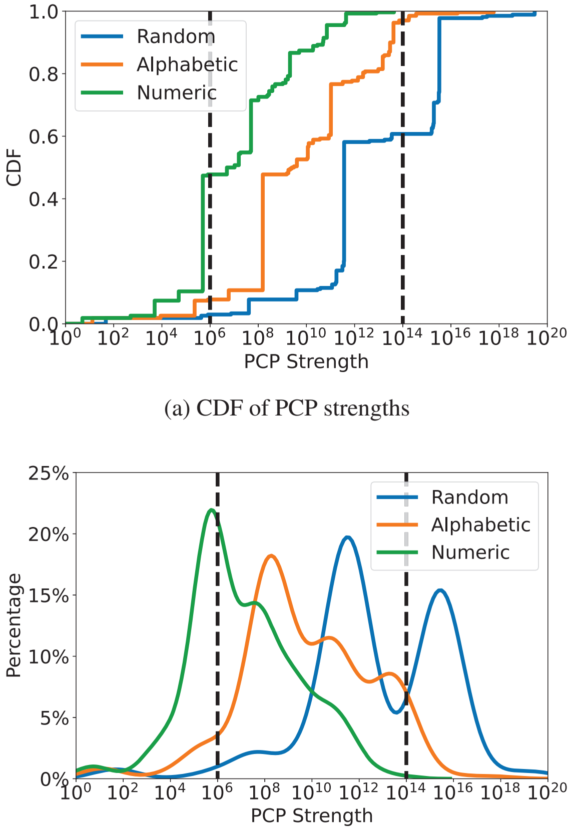
Table 2: Number of PCPs in each category

In addition to estimating PCP strength based on password chosen entirely at random (as is done in previous research [10, 20]), we also consider PCP strength under conditions where users prefer characters from certain character sets: (a) preferring alphabetic (particularly lowercase) characters over non-alphabetic characters (as commonly seen in the US [18]) and (b) preferring numeric characters (as commonly seen in China [18, 36]). These changes help our analysis to more accurately measure the strength of PCPs under a range of usage scenarios. These calculations are performed by modifying our enumeration of password compositions only to include compositions that use the most preferred character classes unless the PCP specifically requires another character class. A more detailed description is given in Appendix B.2.

Throughout our analysis, we categorize PCPs by (i) the country where they are popular, (ii) their Alexa global rank, (iii) their use case, (iv) whether they generate revenue by displaying ads, (v) whether usernames on the website were publicly available or easily guessed, and (vi) whether a data breach had been reported for the website. All categorizations are mutually exclusive, with PCPs popular in multiple countries categorized as “Global”. Table 2 lists these categories and the number of PCPs in each.

7.1 PCP Strength

Figure 1 gives the distribution of password strengths. If passwords are generated entirely at random, nearly all PCPs are strong enough to resist online attacks (10^6 guesses [11]), though only about 40% are strong enough to resist offline



(a) CDF of PCP strengths

(b) Distribution of PCP strengths

10^6 and 10^{14} are estimates of the number of guesses a password should resist to survive online and offline attacks, respectively [11].

Figure 1: PCP Strengths

attacks. For passwords where alphabetic characters are preferred, nearly all PCPs fall into the online-offline chasm [9]—strong enough to resist online attacks but not offline attacks(surviving 10^{14} guesses [11]). This chasm is problematic because PCPs in it impose a usability burden to pick more complex passwords than necessary to resist online attacks, but which are still too weak to resist offline attacks. For passwords where numeric characters are preferred, half of the analyzed PCPs are insufficient to prevent online attacks, and none are strong enough to resist offline attacks.

Comparing mean PCP strength under different password generation strategies, we find that passwords generated at random ($3.5 * 10^{17}$) are roughly two orders of magnitude stronger than alphabetic-preferred passwords ($2.3 * 10^{15}$) and six orders of magnitude stronger than numeric-preferred passwords ($2.1 * 10^{11}$). This highlights the benefits of using a password generator to create passwords. It also demonstrates why it is crucial to consider generation strategy when estimating PCP strength, as assuming passwords are selected entirely at random can significantly overestimate the protectiveness of PCPs.

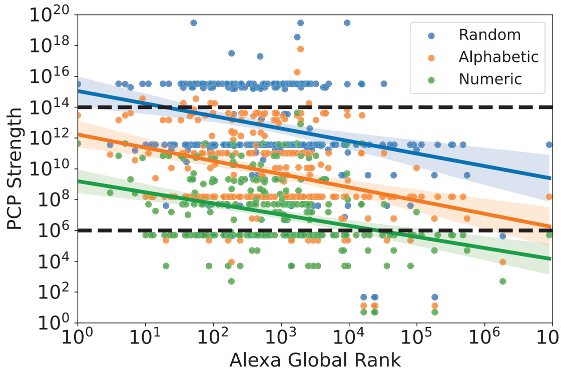


Figure 2: PCP strength by Alexa global rank

7.1.1 Strength by Category

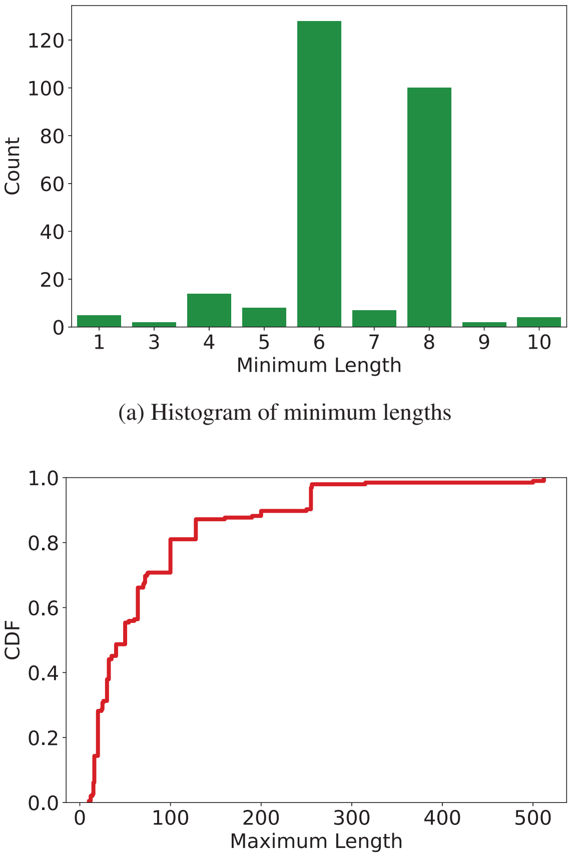
Figure 2 shows the correlation between PCP strength and a website's Alexa global ranking. In general, we find that higher-ranked websites have stronger PCPs. Using Pearson's r and log scales for both rank and PCP strength, we find a medium effect size for entirely random ($r = -0.30$, $p < 0.001$), alphabetic-first ($r = -0.34$, $p < 0.001$), and numeric-first ($r = -0.34$, $p < 0.001$) strengths.

We found a statistically significant difference between strengths based on country for generation at random and alphabetic first generation, but not for numeric-first generation (one-way ANOVA—entirely random—$F(10,259) = 1.87$, $p < 0.05$; alphabetic-first—$F(10,259) = 2.05$, $p < 0.05$; numeric-first—$F(10,259) = 0.29$, $p = 0.98$), We did not find any meaningful pairwise differences for the statistically significant results using Tukey's test. There was no significant difference based on use case (entirely random—$F(5,263) = 1.04$, $p = 0.40$; alphabetic-first—$F(5,263) = 0.59$, $p = 0.74$; numeric-first—$F(5,263) = 0.40$, $p = 0.88$).

Figures showing strength differences based on country, global rank, and use case can be found in Appendix D. We also tested whether (i) ads, (ii) public usernames, (iii) or data breach history impacted PCP strength, finding no statistically significant differences.

7.2 PCP Features

The most common minimum lengths for PCPs are 6 (128; 47%) and 8 (100; 37%) (see Figure 3a). Just over a tenth of PCPs (29; 11%) allowed passwords with fewer than 6 characters, with five (5; 2%) allowing passwords with a single character. These low length requirements are not only problematic for user-generated passwords but also for



(a) Histogram of minimum lengths

(b) CDF of maximum lengths

Figure 3: PCP lengths

password generators, which are known to occasionally generate random but weak passwords at shorter password lengths [24].

Most PCP rules (195; 72%) set a maximum length for passwords, with a wide range of values (see Figure 3b). Just over a tenth (28; 10%) limit passwords to 16 or fewer characters, with four (4; 1%) limited to 12 or fewer characters.

The next most common requirement was having required character classes (51; 19%): digits (42/51; 82%), alphabet (37/51; 73%), lower (12/51; 24%), upper (10/51; 20%), and symbols (4/51; 8%). This was followed by requiring a subset of character classes (43; 16%): at least one (5/43; 12%), two (14/43; 33%), or three (13/43; 30%) characters from all character classes; at least one symbol or digit character (9/43; 21%); at least one upper or symbol character (1/43; 2%); or at least one upper, digit, or symbol character (1/43; 2%).

The remaining requirements only appeared rarely. For prohibited substrings (11; 4%), websites primarily restriction personal information (10/11; 91%): name (6/11; 55%), email (5/11; 45%), username311, birthday211, website name (1/11; 9%). Rules also included max consecutive characters (9; 3%) with values of one (1/9; 11%), two (2/9; 22%), three (4/9; 44%), and seven (1/9; 11%). Finally, one PCP (1; 0%) required two lower case letters and two digits.

7.2.1 Multi-Rule PCPs

Of particular interest, we discovered three PCPs (3; 1%) that had more than one rule.

gumtree.com.au Required twenty-character passwords unless the password included an alphabetic character and either a digit or symbol, in which case ten-character passwords were allowed.

github.com Required fifteen-character passwords unless the password included both a lowercase character and a digit, in which case eight-character passwords were allowed.

yy.com Required nine-character passwords unless the password included an alphabetic character, in which case an eight-character password could be used. This could be to encourage Chinese users to pick non-digit-only passwords, which is common in that culture [18, 36].

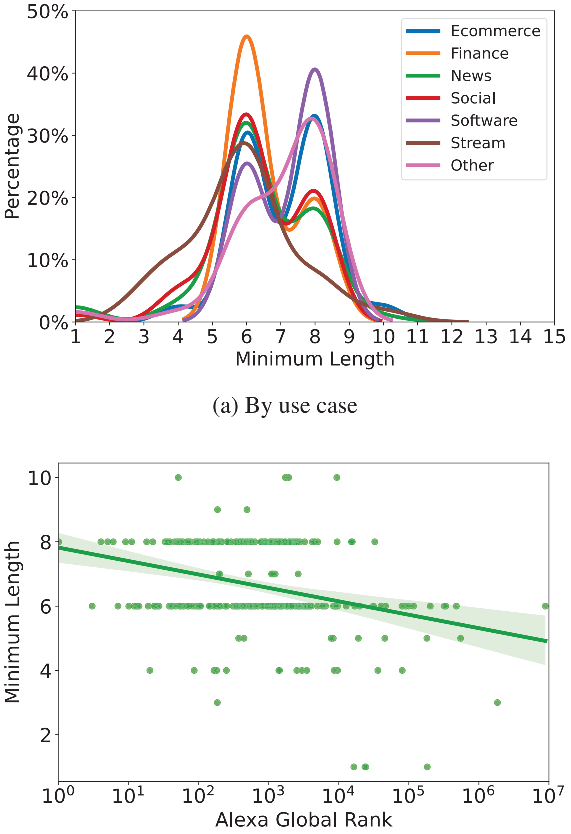
Ignoring specific requirements, these PCPs all share a common goal: allow users to choose between short but complex or long but simple passwords.

7.2.2 Features by Category

We find statistically significant difference for minimum length by country (one-way ANOVA—$F(10,259) = 2.74$, $p < 0.01$), global rank (Pearson's-r—$r = -0.30$, $p < 0.001$), and use case (one-way ANOVA—$F(6,263) = 3.57$, $p < 0.01$). Within these categories, high-ranked websites are much more likely to allow passwords shorter than six characters (see Figure 4b). Similarly, "streaming" websites have lower minimum length requirements (see Figure 4a), with the difference being statistically significant for "Ecommerce" ($p < 0.01$) and "Other" ($p < 0.05$).

We did not find statistically significant differences in maximum length by country (one-way ANOVA—$F(10,259) = 1.05$, $p = 0.40$), global rank (Pearson's-r—$r = 0.01$, $p = 0.88$), or use case (one-way ANOVA—$F(6,263) = 1.40$, $p = 0.21$). We did not see any meaningful difference for other restrictions, though we did not test for statistical significance.

Figures showing differences for minimum and maximum length based on country, global rank, and use case can be found in Appendix E. We also tested whether (i) ads, (ii) public usernames, (iii) or data breach history impacted PCP minimum and maximum length, finding no statistically significant differences.



(a) By use case

(b) By Alexa global rank

Figure 4: PCP minimum lengths

7.3 Website Analysis

We also examined the following items for each website: (a) whether account creation and login required HTTPS, (b) which SSO providers, if any, were supported, and (c) whether a password strength meter is shown to users.

For most websites (255; 94%) HTTPS was required to view the account creation and login pages. Still, there were fifteen (15; 6%) websites where we could access the account creation or login interface over HTTP.[3]

A third of websites (92; 34%) support at least one single sign-on (SSO) provider for account creation and authentication. The most popular SSO providers are Facebook (82/92; 89%), Google (65/92; 71%), Twitter (21/92; 23%), VK (10/92; 11%), and mail.ru (6/92; 7%), with the remaining 20 SSO providers being represented on fewer than five websites.

We find that just over a tenth (35; 13%) of websites show users a strength meter when they are creating passwords. We also find that just under a tenth (22; 8%) use a strength checker as part of their password policy—i.e., passwords must be a certain strength to be accepted.

7.3.1 Websites by Category

For websites whose account creation or login pages can be accessed over HTTP, the majority were in China: China (8/15; 53%), Russia (2/15; 13%), and one each (1/15; 7%) for India, Iran, Nigeria, Brazil, and the US. It is unclear why China is so different, but we find this correlation troubling. These types of websites are most likely to occur in less popular websites.[3]

Within certain countries we see much higher rates of adoption of SSO: Russian (11/19; 58%), Nigeria (6/13; 46%), Brazil (6/14; 43%), Australia (5/13; 38%), UK (3/8; 38%), India (3/9; 33%), Global (21/65; 32%), US (21/72; 29%), China (6/28; 21%), Iran (2/12; 17%). We also see a trend that the less popular sites are more likely to adopt 2FA: Top 10 (1/8; 13%), Top 50 (7/24; 29%), Top 100 (4/25; 16%), Top 500 (17/59; 29%), Top 1000 (6/25; 24%), Top 5000 (37/79; 47%), 5000+ (20/50; 40%). For categories, SSO is more evenly dispersed, though news (35/72; 49%) sites have higher support for SSO.

We do not find any meaningful effect from the categories on strength meters or internal strength checks for passwords.

8 Discussion

In this section, we discuss observations from our research.

8.1 PCP Recommendations

Of all the PCPs encountered in our analysis, we were most interested in the multi-rule PCPs, which allowed users to choose between short but complex or long but simple passwords. This ensures that passwords will resist offline attacks without causing unnecessary usability burdens. Moreover, this approach returns the locus of control to users—i.e., while PCPs are often viewed as restrictive, and therefore less usable [17, 32, 33], multi-rule PCPs give users a choice of which PCP is most appropriate for them. We hypothesize that by giving this control back to users, not only will they be more satisfied with the PCP, but they will also create stronger passwords. Future work could validate this hypothesis and try to determine what the ideal multi-rule construction is. For example, would more rules be even better, providing even more fine-grained control of the types of passwords users can select?

Another observation from our analysis is the importance of PCP design for ensuring the security of passwords not generated entirely at random. Whereas PCP requirements reduce the strength of passwords generated entirely at random (by shrinking the search space), they increase the strength of passwords generated with preferences to a given character class. Thus there is an interesting interplay between PCPs and passwords based on how they are generated. More specifically, we note that increasing length is the easiest way

[3] The list of websites is given in Appendix C.

to improve strength, regardless of generation strategy. Similarly, we find that it is likely advantageous to limit users from having too much of their password be composed of digits (or symbols), as this significantly weakens those passwords and may lead to passwords vulnerable to online guessing attacks. As such, we recommend that administrators use a multi-rule approach that allows users to choose between long but simple passwords or short but complex passwords. This allows machine-generated passwords to be short but ensures that human-generated passwords are strong enough to resist attack.

8.2 NIST Guidelines

NIST provides PCP guidelines (i.e., non-compulsory recommendations) for US companies and organization [12]. While our dataset includes a wealth of PCPs for global and non-US websites, we still think it is interesting to see which of these PCPs conform to the NIST guidelines.

We find that less than half of PCPs (106; 39%) meet NIST's recommended minimum length of eight characters. Similarly, we find that most (195; 72%) implement unnecessary maximum length requirements.

In line with NIST recommendations, most PCPs (177; 66%) do not have any composition requirements (this would be more positive if they met the minimum length requirements). Similarly, only a small fraction (8; 3%) reject specific symbols, which can be an indication of improper password hashing.

9 Related Work

This section discusses related work on password generation, PCP languages, analysis of Web PCPs, and PCP usability.

9.1 PCP Languages

There have been previous proposals for building PCP languages, with each providing a different subset of the features used in our PCP language (see Table 3). Two proposals involve adding additional HTML attributes to input fields to specify PCP requirements [4, 21], though they only cover a small subset of the most common PCP features.

Horsch et al. developed an XML-based PCP language by automatically scanning and extracting PCPs for 72,125 services. Based on a sample of 200 manually verified PCPs, they estimated that their algorithm correctly extracted PCPs in just over four out of five cases, with the remaining cases evenly split between mostly correct and incorrect. Their resulting PCP language has most of the features found in our language. However, it is missing support for multiple rules, requiring a subset of character classes, limiting maximum consecutive characters from the same character class, and set required and prohibited locations based on distance from the end of the password. This demonstrates the limitation of this type of automated PCP extraction—i.e., it can only find PCP features that the automated tool expects to find.

PCP Features	This paper	Daniel Bates [4]	Isiah Meadows [21]	Horsch et al. [14]
Define character sets	✓	✓	✓	✓
Multiple rule sets	✓			
min_length	✓	✓	✓	✓
max_length	✓	✓	✓	✓
max_consecutive	✓	✓		✓
prohibited_substrings	✓		✓	
required	✓	✓	✓	✓
require_subset	✓		✓	
charset_requirements				
.min_required	✓			✓
.max_allowed	✓			✓
.max_consecutive	✓			
.required_locations	✓			✓
.prohibited_locations	✓			✓
reverse indexing	✓			

Table 3: Comparison between PCP languages

Examining our data, none of these PCP languages can encode all the PCPs in our dataset. However, these proposals could be extended to support the features identified in our research. During our PCP language generation process (see §3), our team built and tested several versions of our PCP language that were HTML- and XML-based. Ultimately, we rejected these approaches because our team felt that encoding policies in these languages was cumbersome and that the resulting policies were difficult to read. Still, the results of our user study show that there is significant room for improving our proposed language, and future work could explore integrating paradigms from these prior proposals with our language or testing whether, contrary to our team's perceptions, HTML- or XML-based would be better received than our JSON-based approach by developers. In this regard, the main contribution of our paper is the identification of features that must be included in such PCP languages.

9.2 Web PCP Analysis

In 2010, Florêncio and Herley [10] retrieved PCPs for 75 websites in the US. They found that contrary to their intuition, the importance of a website had little correlation to the PCP used on that website. In many cases, the largest, most important websites had the weakest PCPs. They suggested that the reason for this was that due to market economics, these larger websites needed to be more concerned with usability than security, being able to absorb the security cost of weak PCPs more readily than smaller sites.

In 2016, Mayer et al. [20] replicated and extended the work of Florêncio and Herley. In addition to re-examining 70 of the websites used in the original study (five did not work), they also analyzed 67 German websites. They find that overall, PCP strength has been increasing, though German PCPs, on average, are weaker than US PCPs.

In this paper, we replicate and further extend this work. We collect a dataset that is roughly twice as large and five times more geographically diverse than Mayer et al.'s dataset. Compared to this prior work, we gather more features of the PCPs used on these websites and develop a more fine-grained estimation of PCP strength.

For the most part, our results are similar to past findings. Overall, PCP strength (for random generation) is similar in all studies. However, as our improved strength calculation results in lower estimates of PCP strength, the similarity of our results suggests that PCPs have continued to get stronger over time, though that progress is slow and the delta is not that meaningful. When using PCP strength estimates based on random generation (as the prior work does), we find that PCP strength has become more bimodal, with a clear contrast between websites that require passwords to be offline-resilient and those that only require online-resilience. While this may only be an artifact of our increased precision in plotting PCP strength (the prior worked binned strength into large ranges), we do not believe so and think it is an area that could be explored more in future research. Like the prior work, we find that most PCPs reside within the online-offline chasm identified by Florêncio and Herley [11].

Like prior work, we find no statistically significant correlations when comparing PCP strength based on country, use case, public usernames, and past breaches. However, unlike the prior work, we find a correlation between a website's popularity and the strength of its PCPs. This difference is most likely explained by (a) our larger data set, (b) the increased fidelity of our strength estimates, and (c) the use of log adjusted strength and global ranks. Also, whereas prior work found a negative correlation between whether a website served ads and its PCP strength, we find no statistically significant correlation.

9.3 PCP Usability

Several studies have examined the effect of password policies on user behavior. These studies have shown that while strong PCPs make passwords harder to crack, they also make passwords harder for users to remember [29]. Furthermore, as the number of passwords a user needs increases, their ability to remember them decreases [1, 35]. This helps explain why when Florêncio and Herley [9] studied password behavior of half a million users, they found that users had on average 25 passwords and reused any given password on an average of 6.5 different websites.

Other research explores what PCP features make passwords harder to remember, with most research finding that it is complex character class requirements that cause the most difficulty [17, 32, 33]. In contrast, minimum length is not nearly as significant of an impediment, leading researchers to suggest favoring longer but less complicated passwords. More recently, we have seen these suggestions reflected in NIST guidelines [12].

Our research finds that length has the greatest impact on PCP strength for both passwords generated at random and using an alphabetic-first approach. As such, we echo prior recommendations for PCPs to focus on length as opposed to complexity. For those that want the best of both worlds, multi-rule PCPs can be used that allow short but complex or long but simple passwords, giving users the locus of control for this decision and thereby increasing usability. Similarly, due to the weaknesses of digit-first generated passwords, PCPs should likely restrict the usage of too many digits in a password.

10 Conclusion and Future Work

In this work, we developed a PCP language that websites and password managers can use to support the generation of compliant passwords. We hope that our work will signal to both communities that adopting a PCP language has tangible benefits. For websites, it allows them to unify their PCP specification and checking, allowing changes to the PCP file to automatically update how checking happens on both the client and server. For password managers, it not only improves the usability and utility of password management but also supports opinionated generation algorithms (e.g., mobile-aware generation [13], security-focused generation [24]), which would otherwise frequently generate non-compliant passwords.

While we are encouraged by the positive results of our user study, they also indicated that there is room for improvements. Future work could expand our PCP language by identifying and adding support for rarely used PCP features, such as restricting sequences of characters (e.g., "abcde") or keyboard patterns (e.g., "qwerty"). Similarly, our language could be enhanced to allow Unicode characters. Future research could also examine how to allow our PCP language to handle dynamic strings (e.g., usernames). One potential solution is to use placeholders in the *prohibited_substrings* requirement, providing appropriate values to the library at password validation. Finally, research could explore automatically identifying PCPs, both in whitebox scenarios, helping web developers identify their website's PCP, and blackbox scenarios, helping password managers identify PCPs for websites that do not publish it, with care taken to avoid flooding servers with passwords guesses (approximating a DoS attack).

References

[1] Anne Adams and Martina Angela Sasse. Users are not the enemy. *Communications of the ACM*, 42(12):40–46, 1999.

[2] Khaldoun M Aldiabat and Carole-Lynne Le Navenec. Data saturation: The mysterious step in grounded theory methodology. *The Qualitative Report*, 23(1):245–261, 2018.

[3] Aaron Bangor, Philip T Kortum, and James T Miller. An empirical evaluation of the system usability scale. *Intl. Journal of Human–Computer Interaction*, 24(6):574–594, 2008.

[4] Daniel Bates. Proposal: Html passwordrules attribute. https://github.com/whatwg/html/issues/3518, 2021.

[5] Joseph Bonneau, Cormac Herley, Paul C Van Oorschot, and Frank Stajano. The quest to replace passwords: A framework for comparative evaluation of web authentication schemes. In *2012 IEEE Symposium on Security and Privacy*, pages 553–567. IEEE, 2012.

[6] John Brooke. Sus: a "quick and dirty'usability. *Usability evaluation in industry*, 189(3), 1996.

[7] Anupam Das, Joseph Bonneau, Matthew Caesar, Nikita Borisov, and XiaoFeng Wang. The Tangled Web of Password Reuse. In *Network and Distributed System Security (NDSS)*, volume 14, pages 23–26, 2014.

[8] Matteo Dell'Amico, Pietro Michiardi, and Yves Roudier. Password strength: An empirical analysis. In *2010 Proceedings IEEE INFOCOM*, pages 1–9. IEEE, 2010.

[9] Dinei Florencio and Cormac Herley. A large-scale study of web password habits. In *Proceedings of the 16th international conference on World Wide Web*, pages 657–666, 2007.

[10] Dinei Florêncio and Cormac Herley. Where do security policies come from? In *Proceedings of the Sixth Symposium on Usable Privacy and Security*, pages 1–14, 2010.

[11] Dinei Florêncio, Cormac Herley, and Paul C Van Oorschot. An administrator's guide to internet password research. In *28th Large Installation System Administration Conference (LISA14)*, pages 44–61, 2014.

[12] Paul A Grassi, James L Fenton, Elaine M Newton, Ray A Perlner, Andrew R Regenscheid, William E Burr, Justin P Richer, Naomi B Lefkovitz, Jamie M Danker, YeeYin Choong, et al. Nist special publication 800-63b: Digital identity guidelines. *National Institute of Standards and Technology (NIST)*, 27, 2016.

[13] Kristen K Greene, John Michael Kelsey, Joshua M Franklin, et al. *Measuring the usability and security of permuted passwords on mobile platforms*. US Department of Commerce, National Institute of Standards and Technology, 2016.

[14] Moritz Horsch, Mario Schlipf, Johannes Braun, and Johannes Buchmann. Password requirements markup language. In *Australasian Conference on Information Security and Privacy*, pages 426–439. Springer, 2016.

[15] Freedom House. Freedom house (fh) freedom of the press report. https://freedomhouse.org/reports/publication-archives.

[16] N. Huaman, S. Amft, M. Oltrogge, Y. Acar, and S. Fahl. They would do better if they worked together: The case of interaction problems between password managers and websites. In *2021 2021 IEEE Symposium on Security and Privacy (SP)*, pages 1626–1640, Los Alamitos, CA, USA, may 2021. IEEE Computer Society.

[17] Saranga Komanduri, Richard Shay, Patrick Gage Kelley, Michelle L Mazurek, Lujo Bauer, Nicolas Christin, Lorrie Faith Cranor, and Serge Egelman. Of passwords and people: measuring the effect of password-composition policies. In *Proceedings of the sigchi conference on human factors in computing systems*, pages 2595–2604, 2011.

[18] Zhigong Li, Weili Han, and Wenyuan Xu. A large-scale empirical analysis of chinese web passwords. In *23rd USENIX Security Symposium (USENIX Security 14)*, pages 559–574, San Diego, CA, August 2014. USENIX Association.

[19] Sanam Ghorbani Lyastani, Michael Schilling, Sascha Fahl, Michael Backes, and Sven Bugiel. Better managed than memorized? studying the impact of managers on password strength and reuse. In *27th {USENIX} Security Symposium ({USENIX} Security 18)*, pages 203–220, 2018.

[20] Peter Mayer, Jan Kirchner, and Melanie Volkamer. A second look at password composition policies in the wild: Comparing samples from 2010 and 2016. In *Thirteenth Symposium on Usable Privacy and Security (SOUPS 2017)*, pages 13–28, Santa Clara, CA, July 2017. USENIX Association.

[21] Isiah Meadows. Add password restriction attributes. https://discourse.wicg.io/t/add-password-restriction-attributes-to-input-type-password/4767, Sep 2020.

[22] Alena Naiakshina, Anastasia Danilova, Eva Gerlitz, and Matthew Smith. On conducting security developer studies with cs students: Examining a password-storage study with cs students, freelancers, and company developers. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, pages 1–13, 2020.

[23] Alena Naiakshina, Anastasia Danilova, Eva Gerlitz, Emanuel Von Zezschwitz, and Matthew Smith. " if you want, i can store the encrypted password" a password-storage field study with freelance developers. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, pages 1–12, 2019.

[24] Sean Oesch and Scott Ruoti. That was then, this is now: A security evaluation of password generation, storage, and autofill in browser-based password managers. In *USENIX Security Symposium*, 2020.

[25] Timothy Oesch. *An Analysis of Modern Password Manager Security and Usage on Desktop and Mobile Devices*. PhD thesis, The University of Tennessee, 2021.

[26] OWASP. Password special characters. https://owasp.org/www-community/password-special-characters, 2021.

[27] Sarah Pearman, Jeremy Thomas, Pardis Emami Naeini, Hana Habib, Lujo Bauer, Nicolas Christin, Lorrie Faith Cranor, Serge Egelman, and Alain Forget. Let's go in for a closer look: Observing passwords in their natural habitat. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 295–310. ACM, 2017.

[28] Sarah Pearman, Shikun Aerin Zhang, Lujo Bauer, Nicolas Christin, and Lorrie Faith Cranor. Why people (don't) use password managers effectively. In *Fifteenth Symposium On Usable Privacy and Security (SOUPS 2019). USENIX Association, Santa Clara, CA*, pages 319–338, 2019.

[29] Robert W Proctor, Mei-Ching Lien, Kim-Phuong L Vu, E Eugene Schultz, and Gavriel Salvendy. Improving computer security for authentication of users: Influence of proactive password restrictions. *Behavior Research Methods, Instruments, & Computers*, 34(2):163–169, 2002.

[30] Shannon Riley. Password security: What users know and what they actually do. *Usability News*, 8(1):2833–2836, 2006.

[31] Jeff Sauro and James R Lewis. *Quantifying the user experience: Practical statistics for user research*. Morgan Kaufmann, 2016.

[32] Richard Shay, Saranga Komanduri, Adam L Durity, Phillip Huh, Michelle L Mazurek, Sean M Segreti, Blase Ur, Lujo Bauer, Nicolas Christin, and Lorrie Faith Cranor. Can long passwords be secure and usable? In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2927–2936, 2014.

[33] Richard Shay, Saranga Komanduri, Adam L Durity, Phillip Huh, Michelle L Mazurek, Sean M Segreti, Blase Ur, Lujo Bauer, Nicolas Christin, and Lorrie Faith Cranor. Designing password policies for strength and usability. *ACM Transactions on Information and System Security (TISSEC)*, 18(4):1–34, 2016.

[34] Blase Ur, Fumiko Noma, Jonathan Bees, Sean M Segreti, Richard Shay, Lujo Bauer, Nicolas Christin, and Lorrie Faith Cranor. 'I added '!' at the end to make it secure': Observing Password Creation in the Lab. In *Proceedings of the Eleventh Symposium On Usable Privacy and Security*, 2015.

[35] Kim-Phuong L Vu, Robert W Proctor, Abhilasha Bhargav-Spantzel, Bik-Lam Belin Tai, Joshua Cook, and E Eugene Schultz. Improving password security and memorability to protect personal and organizational information. *International Journal of Human-Computer Studies*, 65(8):744–757, 2007.

[36] Ding Wang, Ping Wang, Debiao He, and Yuan Tian. Birthday, name and bifacial-security: understanding passwords of chinese web users. In *28th USENIX Security Symposium (USENIX Security 19)*, pages 1537–1555, 2019.

[37] Ke Coby Wang and Michael K Reiter. How to end password reuse on the web. *arXiv preprint arXiv:1805.00566*, 2018.

A Study Instrument

Setup For this study, you will be using a python library we developed. Please install this library using pip: `python3 -m pip install -user -upgrade password-policy`. If for some reason, you don't have pip installed, you can install it using: `python3 -m ensurepip -user -upgrade`.

After installation is complete, check that everything is working correctly by copying and pasting the following command into your terminal. Enter the resulting output below: `python3 -c "import password_policy; print(password_policy.__version__)"`.

Q1. Enter version

Demographics
Q2.1. What is your class standing?
○ *Junior* ○ *Senior* ○ *MS student* ○ *PhD student*

Q2.2. What is your major?
○ *Computer Science* ○ *Computer Engineering* ○ *Electrical Engineering* ○ *other [Enter here]*

Q2.3. What is your sex?
○ *Male* ○ *Female* ○ *Non-binary* ○ *Prefer not to answer*

Tasks Different websites have different requirements for passwords. For example, some websites may require passwords to have a minimum length, include certain types of characters, and avoid using other characters. In our research group, we are studying a system for describing password policies using JSON. We are also studying libraries that can be used to construct these JSON policy descriptions and validate passwords based on these descriptions.

In this study, you will use this system and a python library to encode several password policies. Our goal is to understand how usable this system and library is.

To help you learn about this system and the python library you installed, please click [this link to view the relevant documentation]. You will be using the knowledge from this documentation for the rest of the study. Feel free to refer to it throughout the study. A link to this documentation will always be available on the pages describing your tasks for this study.

When you feel ready to use this system, click continue to be given your first task.

The following questions were the same for each policy, except for the policy requirements. We give the full text for Policy 1's questions, and only the policy requirements for Policy 2–5.

Q3.1.1. Using the python library, please write a policy description for the following password policy. When finished, encode it in JSON and enter it into the text field below. We will validate the entered policy description to make sure it is correct. You may also directly write the policy as JSON (not using the library) if desired.

Password policy:

- The password must be at least 8 characters long

[Documentation link]

Q3.1.2. Did you manually write the JSON policy description, or did you generate it using the python library?
○ *Generated it using Python library* ○ *Manually entered the JSON policy*

Q3.1.3. Based on your experience authoring the JSON policy description, indicate to what extent you agree with the following statements. Options:Strongly disagree-1..Strongly agree-7
○ *Overall, I am satisfied with the ease of completing this task.* ○ *Overall, I am satisfied with the amount of time it took to complete this task.* ○ *Overall, I am satisfied with the support information (on-line help, messages, documentation) when completing this task.*

Q3.2.1. Password policy:

- The password must be at least 8 characters long
- The password must contain characters from at least two of the following: uppercase letters, lowercase letters, numbers, symbols

Q3.3.1. Password policy:

- The password must be at least 12 characters long
- The password must contain at least one letter and one number
- The password must NOT contain space

Q3.4.1. Password policy:

- The password must satisfy one OR the other of the following policies:
    - The password must be at at least 8 characters long
    - The password must contain at least one letter and one number
- OR
    - The password must be at least 15 characters long

Q3.5.1. Password policy:

- The password must be at least 8 characters long
- The password must contain at least two symbols
- The password must contain at least one uppercase letter and one lowercase letter
- The password must NOT contain space, the carrot symbol (ˆ), quotes ('), double quotes ("), semicolons (;), slashes (/), or backslashes (\).
- The password must NOT contain the substring "mywebsite"

Post-Study Questionnaire That was the last policy you will need to write for this study. We will now ask you a few questions about your experience the password policy description system and python library.

Q4.1. Please answer the following question about your experience. Try to give your immediate reaction to each statement without pausing to think for a long time. Mark the middle column if you don't have a response to a particular statement.

Options: Strongly Disagree, Disagree, Neither Agree nor Disagree, Agree, Strong Agree

1. I think that I would like to use this system frequently
2. I found the system unnecessarily complex
3. I thought the system was easy to use
4. I think that I would need the support of a technical person to be able to use this system
5. I found the various functions in this system were well integrated
6. I thought there was too much inconsistency in this system
7. I would imagine that most people would learn to use this system very quickly
8. I found the system very cumbersome to use
9. I felt very confident using the system
10. I needed to learn a lot of things before I could get going with this system

Q4.2. What did you like the most about the system and library?

Q4.3. What did you like the least about the system and library?

Q4.4. Is there any other feedback you would like us to know about the system or library?

B PCP Strength Calculations

We measure the strength of password composition policies (PCPs) by estimating how many passwords exist that (a) satisfy the PCP and (b) are of the shortest possible length. We then divide this number by two to estimate the average number of guesses an adversary needs to find a user's passwords. This approach gives an exact estimate of strength when passwords are generated entirely at random. To estimate strength for human-generated passwords, we allow our strength estimates to be parameterized by what character classes are preferred [18].

B.1 Algorithm

Step 1—Preprocessing First, we filter the rules and only consider those that have the smallest *min_length* (there may be multiple). Next, we simplify *require_subset*, creating a new rule with *require* set for each possible combination of the listed *options* of length *count*. Lastly, we simplify the shortcut rules *require*, setting *min_require* for each charset listed in the requirement.

Step 2—Enumerating Password Compositions In this step, we enumerate all possible password compositions for the rules identified in Step 1. A password composition is simply a list specifying how many characters from each character class are used to make up a password. For example, for a PCP that (a) only allows lowercase letters and digits and (b) has a rule that sets *min_length* to 2 (but no other requirements), there are three password compositions: (1) two lowercase letters, (2) two digits, (3) one lowercase letter and one digit. Note, we only consider compositions where the sum of character counts equals *min_length*.

We take the following steps to derive the password compositions for a rule. First, we create a password composition with values set based on *min_required* for each charset. We also calculate *required_chars*, which tracks the total number of required characters (sum of the calculated password composition). Second, we create a list of length *min_length* − *required_chars*. At each index *i* (one-indexed) of this new list, we include a list of which character classes could appear *i* more times in the password composition without violating *max_allowed* for each charset (if set). Third, we calculate the full factorial combination of items in this list of lists. For each such combination, we create a new password composition that takes the original password combination and adds the character classes in the combination. For each composition, we also store any restrictions related to that composition that may not yet have been handled (e.g., *max_consecutive*).

For example, consider a policy with *min_length* set to 3, which requires the *alphabet* character set to be used once and has at most one digit. Our initial password composition would be *[1,0,0]* representing 1 alphabet character, 0 digits, and 0 symbols; *required_characters* would be 1. Our list of lists would be *[[alphabet,digit,symbol], [alphabet,symbol]]*. In total, there are six ($3*2$) possible combinations of this list, which after added to initial password composition result give the following password compositions: *[[3,0,0], [2,0,1], [2,1,0], [1,1,1], [2,0,1], [1,0,2]]*.

This method will not result in overlapping compositions within a given rule but can between rules. If this occurs, duplicate compositions are trimmed.

Step 3—Calculating Combinations and Permutations For each composition, we will calculate the number of passwords (i.e., size of the search space) represented by each composition that also satisfy the PCP. As a password only maps to a single composition, the sum of search space sizes for each composition is the size of the overall password search space. For each composition, we take the following steps to calculate its search space size:

We start by calculating the number of combinations of characters from the charsets that make up the composition:

$$\prod_i charset\_size_i^{composition_i} \tag{1}$$

We then multiply this value by the number of unique permutations in the composition:

$$\frac{(\sum_i composition_i)!}{\prod_i (composition_i!)} \tag{2}$$

If there are no additional requirements to be considered, this value is used as the composition's search space size. If there are additional requirements, we will reduce this calculated by value by the number of passwords removed by each requirement.

First, we consider the *required_locations* requirement. If used, we recalculate our baseline using the same calculations above, except that we reduce the permutation calculation to only consider character classes not at fixed positions due to *required_locations*.

For the remaining four requirements, we take an approach wherein we create one or more invalid compositions that violate the requirement, calculate the search space for the invalid composition, and subtract the invalid composition's search space size from the overall composition's search space size (calculated above). We continue doing so until there are no more requirements to handle. We generate these invalid compositions as follows:

- For *max_consecutive*, we identify all charsets which have enough occurrences in the composition to violate this rule. For each of these charsets, we create a new, invalid composition that removes ($max\_consecutive + 1$) occurrences from matched charset and adds a single occurrence of a new charset of size equal to the matched charset (representing the repeated character).
- For *max_consecutive* in *charset_restrictions*, we do much the same as above, except that the size of the new charset in the invalid compositions will equal $matched\_charset\_size^{max\_consecutive+1}$, representing all possible combinations of the charset.
- For each substring in *prohibited_substrings*, we create a new, invalid composition that removes the appropriate charset for each character in the substring. We then append a charset of size 1 to the composition, representing the prohibited string.
- For each location in *prohibited_location*, we do not modify the current composition but instead calculate its search space as if the prohibited location were required.

B.2 Estimating Human-Generation

Prior research has shown that when generating passwords, humans prefer characters from specific character classes, though this preference can differ based on country [18, 36]. Our PCP strength estimation can be parameterized based on what character classes users prefer to represent this behavior. For example, American users' preferences might be lowercase, uppercase, then digits [18]. For Chinese users, their preferences are more likely to be digits, lowercase, then uppercase [18, 36].

We handle these preferences in Step 2 of our calculations. We initially execute step two as described up through calculating the list of lists representing characters that can occur in the remaining spots of the initially calculated password composition. For each sublist of charsets, we check to see if any of those charsets appears in the list of preferred charsets. If one or more do, we replace the sublist with a new list with a single element matching the highest-ranked matching charsets. After this modification, calculations proceed as described.

Note, these preference-based calculations are Fermi approximations, underestimating character class diversity in user passwords and overestimating diversity of character selection within a character class, with the two errors hopefully canceling out. Even though these are not exact estimates for human-generated passwords, they are sufficient to help administrators and researchers estimate the overall strengths and weaknesses of a PCP.

B.3 Limitations

For PCPs that do not use any of the final four requirements discussed in Step 3, our method precisely calculates the PCP's search space. Our calculation is also correct if only a single one of these requirements are used for a composition. Of the 270 PCPs in our dataset, 260 do not use any of the five requirements, and of the ten that do, each uses only a single requirements. This means that calculations used in our analysis are all precise, and it suggests that most PCPs will have their search space calculated precisely.

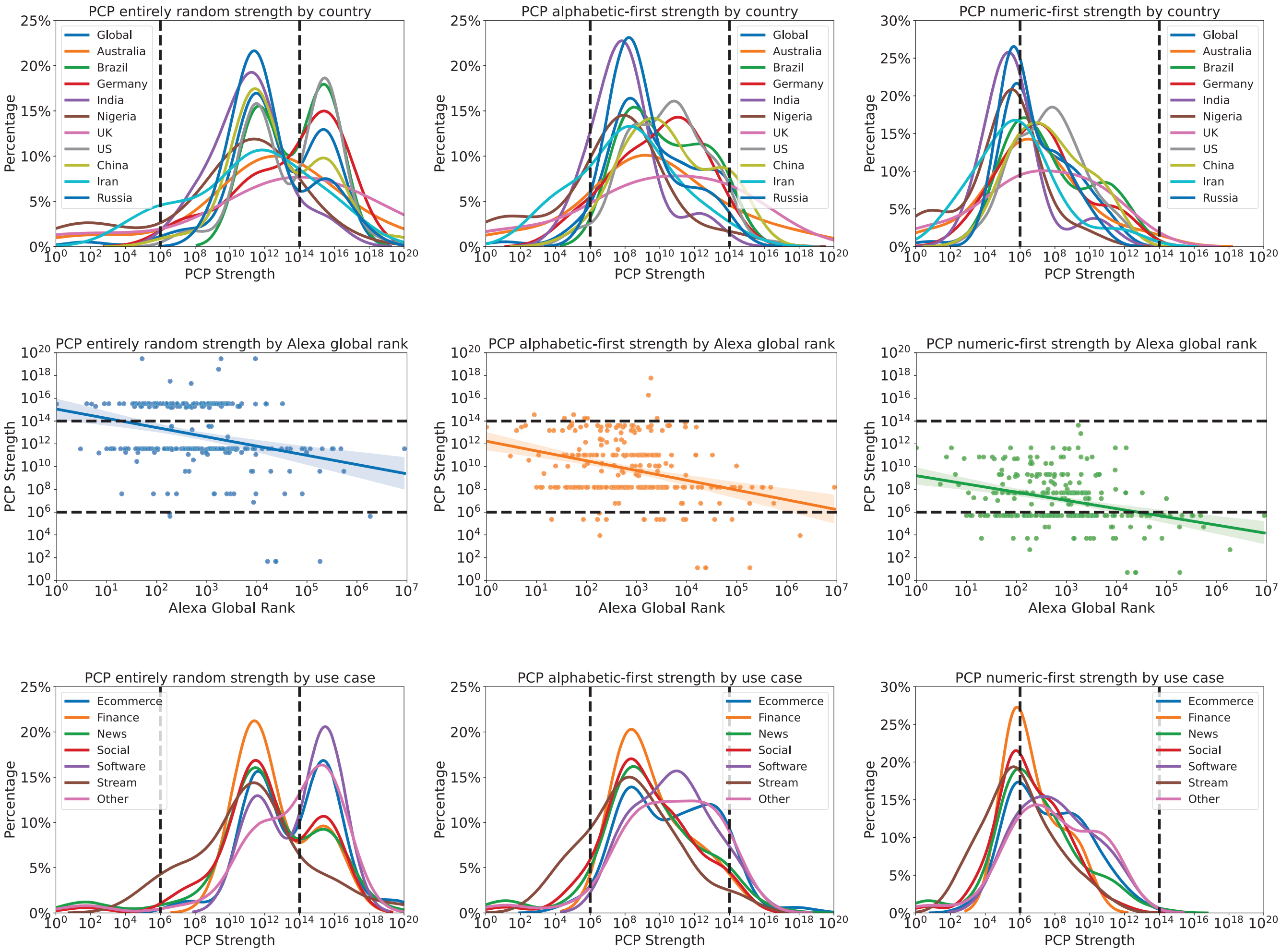
Still, more complicated PCPs that use multiple of the five requirements could have their search spaces underestimated. This occurs because these requirements have the possibility of removing the same passwords. To our knowledge, the only way to prevent this would be to enumerate the password combinations and permutations–as we did with compositions—but this is not tractable for any meaningful length of passwords. However, as the reduction to the search space for each of these requirements will generally be small compared to the overall size of the composition's search space, we believe that the underestimates should be minimal. Additionally, in terms of strength estimates, underestimates are safer than overestimates.

C Webpages Accessible with HTTP

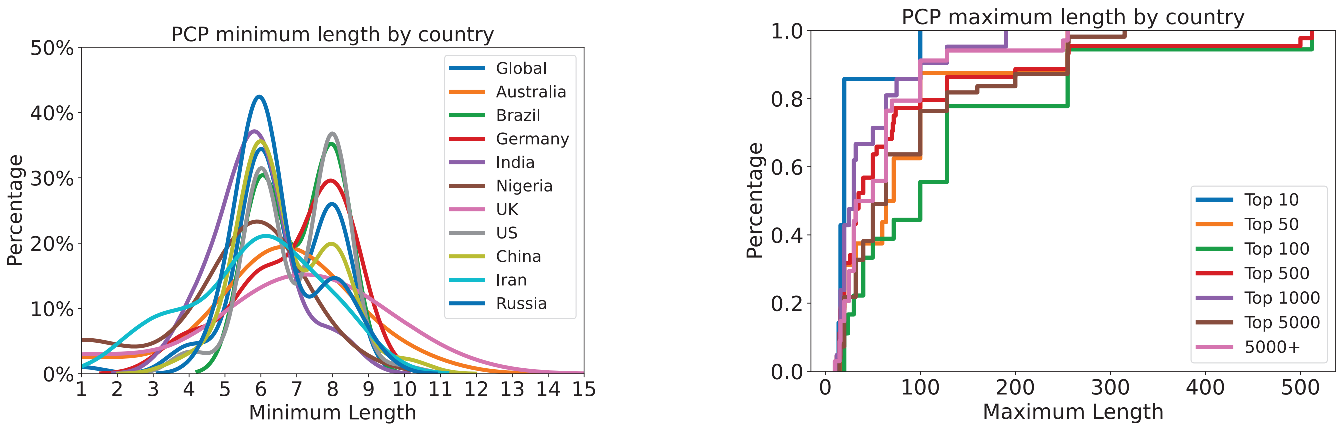
The following is the list of website for which we were able to access the account creation or login page using HTTP:

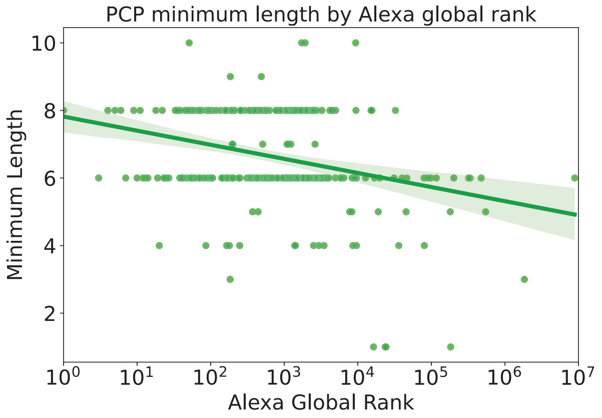
Website	Country	Popularity	Category
weibo.com	China	Top 50	Social
babytree.com	China	Top 100	Social
usatoday.com	US	Top 500	News
yaplakal.com	Russia	Top 5000	Social
ig.com.br	Brazil	Top 5000	Social
wikidot.com	China	Top 5000	Other
fb.ru	Russia	Top 5000	News
javlibrary.com	China	5000+	Stream
dwnews.com	China	5000+	News
metacafe.com	India	5000+	Social
eskimi.com	Nigeria	5000+	Social
ci123.com	China	5000+	Stream
sinovision.net	China	5000+	News
sugardaddyforme.com	China	5000+	Social
mydiba.xyz	Iran	5000+	Stream

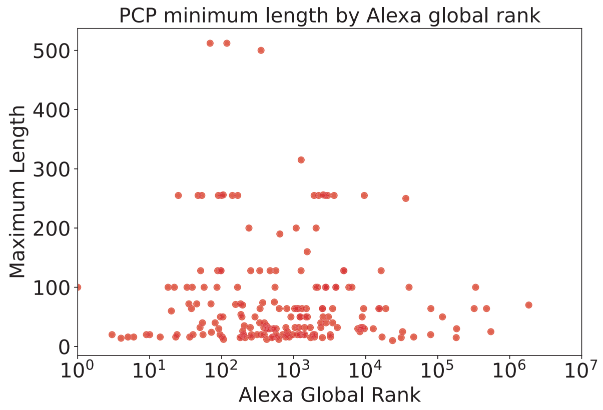
D PCP Strength By Category

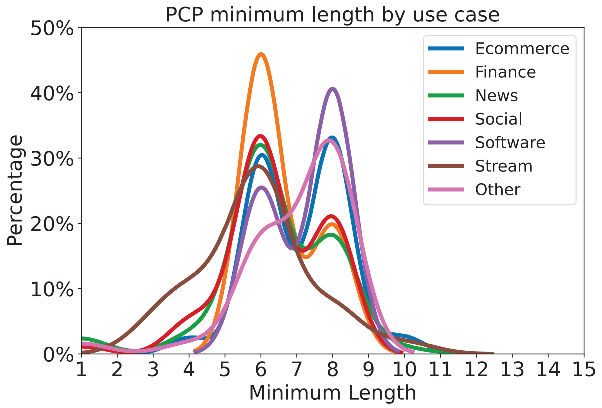


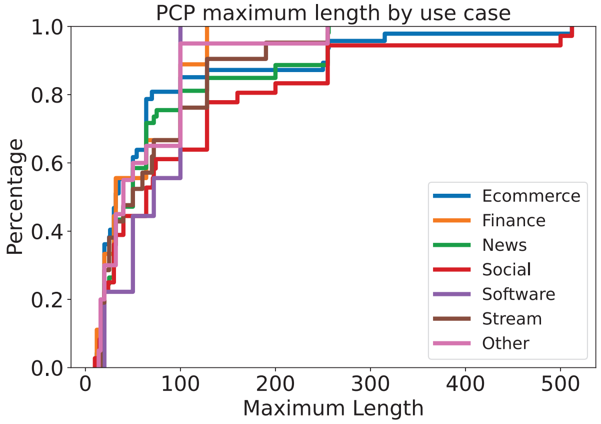
E PCP Features by Category




PCP minimum length by Alexa global rank


PCP minimum length by Alexa global rank


PCP minimum length by use case


PCP maximum length by use case