Skeptic: Automatic, Justified and Privacy-Preserving Password Composition Policy Selection

Saul Johnson
saul.johnson@tees.ac.uk
Data Analytics Research Group
Teesside University
Middlesbrough, UK

João F. Ferreira
joao@joaoff.com
INESC-ID and Instituto Superior Técnico
University of Lisbon
Lisbon, Portugal

Alexandra Mendes
alexandra@archimendes.com
HASLab, INESC TEC, Porto and
Department of Informatics
Universidade da Beira Interior
Covilhã, Portugal

Julien Cordry
j.cordry@tees.ac.uk
Immersive Tech. Research Group
Teesside University
Middlesbrough, UK

ABSTRACT

The choice of password composition policy to enforce on a password-protected system represents a critical security decision, and has been shown to significantly affect the vulnerability of user-chosen passwords to guessing attacks. In practice, however, this choice is not usually rigorous or justifiable, with a tendency for system administrators to choose password composition policies based on intuition alone. In this work, we propose a novel methodology that draws on password probability distributions constructed from large sets of real-world password data which have been filtered according to various password composition policies. Password probabilities are then redistributed to simulate different user password reselection behaviours in order to automatically determine the password composition policy that will induce the distribution of user-chosen passwords with the greatest uniformity, a metric which we show to be a useful proxy to measure overall resistance to password guessing attacks. Further, we show that by fitting power-law equations to the password probability distributions we generate, we can justify our choice of password composition policy without any direct access to user password data. Finally, we present Skeptic—a software toolkit that implements this methodology, including a DSL to enable system administrators with no background in password security to compare and rank password composition policies without resorting to expensive and time-consuming user studies. Drawing on 205,176,321 passwords across 3 datasets, we lend validity to our approach by demonstrating that the results we obtain align closely with findings from a previous empirical study into password composition policy effectiveness.

CCS CONCEPTS

• **Security and privacy** → **Formal security models**; **Logic and verification**; *Authentication*; *Systems security*.

KEYWORDS

Password composition policy, Passwords, Password authentication, Formal verification, Interactive theorem proving

ACM Reference Format:
Saul Johnson, João F. Ferreira, Alexandra Mendes, and Julien Cordry. 2020. Skeptic: Automatic, Justified and Privacy-Preserving Password Composition Policy Selection. In *Proceedings of the 15th ACM Asia Conference on Computer and Communications Security (ASIA CCS '20), October 5–9, 2020, Taipei, Taiwan.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3320269.3384762

1 INTRODUCTION

If we wish to increase the resilience of a password-protected system to password guessing attacks, our thoughts might turn in the first instance to password composition policies—sets of rules that dictate which subset of the space of all supported passwords users are permitted to create on the system. The selection of a suitable password composition policy, however, has not historically been carried out according to rigorous selection criteria [8], with a tendency for system administrators to base their decision on which one "feels" like it would lead to a more secure distribution of user-chosen passwords. For such a critical component of system security, the finding that the restrictiveness of a password composition policy has little to no correlation with the value of the assets it protects is somewhat alarming [14], and makes a strong case for a more rigorous method of selection.

While much study to date has been conducted on how password composition policies affect the security of password-protected systems, such work usually consists of an analysis of either leaked data sets that have since been released into the public arena [36] or of passwords that have been collected under different password composition policies specifically for the purpose of the study [24, 31]. The former condition means that it is very difficult to estimate how some of the more exotic password composition policies affect system security because databases of passwords created under those policies are not available. While it might be tempting to merely filter these datasets according to the policy we wish to examine, previous work [22] finds that this does not create a dataset that

ASIA CCS '20, October 5–9, 2020, Taipei, Taiwan

is representative of one that is actually created under that policy, with passwords in filtered datasets tending to be stronger. The latter condition, while allowing password security researchers to collect data under any password composition policy they choose, has considerably less ecological validity; the participants were, after all, creating passwords in an experimental setting and not on any real-world system of value to them as individuals. Gathering and analysing data in this way is also expensive, time-consuming, and requires significant domain expertise, placing it beyond the reach of a typical system administrator working in the field. Finally, both of these methodologies raise privacy concerns. In each case, we are handling user-generated passwords that may still be in use by those individuals, or else be usable to infer passwords that are. We are motivated, therefore, to search for a methodology that permits us to automatically choose a suitable password composition in a way that allows us to justify that choice while avoiding the propagation of the user password data that informs it. This is especially important considering the recent rise in previously-leaked passwords being employed in phishing scams [29] against the users they belong to.

In this work, we propose such a methodology, and present Skeptic—a software toolchain that puts it into practice. We begin by drawing on large sets of leaked password data [6, 10, 18] to derive password probability distributions. By redistributing password probabilities in different ways, we can simulate different modes of password reselection behaviour that might be exhibited by users when forced to select a different password by the password composition policy. Drawing on work by Malone and Maher [26] and Wang et al. [35], we fit power-law curves to these password probability distributions, allowing us to quantify the additional guessing attack resistance conferred by their associated password composition policies in isolation from the password data itself. Following related research into increasing system security by maximising password diversity [4, 26, 30], we achieve this by using the *uniformity* of these distributions as a proxy for their overall resistance to guessing attacks. To maximise the practical utility of the data we generate, Skeptic includes the Password Composition Policy Assertion Language (Pacpal)—a DSL for straightforwardly comparing and ranking password composition policies using this data.

Using a selection of password composition policies drawn from related work and data from three large-scale password data breaches, we demonstrate our methodology and its implementation (as the Skeptic toolchain) by rigorously and justifiably ranking password composition policies under a range of different assumptions about user password reselection behaviour. As our evaluation data, we use 3 datasets containing a total of 205,176,321 passwords, studying 28 distinct password composition policies. The results we obtain correlate strongly with those from previous empirical studies on the effects of password composition policies on the security of user-chosen passwords, with some interesting findings that warrant further study. For instance, we find that stricter (i.e. less usable) password composition policies dramatically reduce password probability distribution uniformity if we assume that user password reselection behaviour will converge on a small number of remaining permitted passwords. We further demonstrate that the Skeptic toolchain supports straightforward specification of password composition policies from within the *Coq* proof assistant, with all the advantages we would expect from such an encoding, including the ability to check from within *Coq* that certain password composition policies confer immunity to the *Mirai* and *Conficker* botnet malware.

We have introduced the work and its motivation in this Section 1. In Section 2, we introduce related work. We then move on to describing our methodology in detail in Section 3, in a manner designed to facilitate implementation to encourage replication and experimentation. In Section 4 we describe the implementation of our methodology as the Skeptic toolchain. Section 5 contains an evaluation of our approach, in which we attempt to replicate previous empirical research [31] on password composition policy effectiveness. Finally, we conclude in Section 6.

2 RELATED WORK

There exists a wealth of password data online that has been compromised from various sources and released into the public arena. Weir et al. [36] draw on a few different sets of this data in order to examine the validity of using password entropy as defined in NIST document SP800-63-1 [8] to determine the security provided by various password composition policies. The authors conclude, based on experiments run against some of the same datasets we use in this work [11], that it is not a valid metric, empirically validating earlier work by Verheul [34] proving that conversion of Shannon entropy-like measures into password guessing entropy under different password composition policies is not possible. Such work demonstrates the effective use of large breached datasets in password composition policy research, and in Section 5.3 we replicate a subset of its results as part of validation of our novel methodology.

It is also possible to use these breached user credential databases straight away to inform our choice of password policy by simply prohibiting all the passwords we can find in them outright. The *Pwned Passwords* web application and API [19] provides this functionality as a service, aggregating over 500 million unique passwords that have been exposed in data breaches and made publicly available online. Just because a password has not been exposed before, however, does not mean that it is a good password. At the time of writing, for instance, *"breakfast321"* is not present in *Pwned Passwords* but as a dictionary word and run of sequential digits is very likely to be cracked with minimal effort by any of the great number of password cracking algorithms in widespread use today [37, 39], with the popular *zxcvbn* password strength checking library [38] estimating that this particular example could be cracked in around 10^5 guesses—well within the capabilities of even the lowliest attacker. The inadequacy of blacklist-based measures alone motivates work such as ours, which aims to equip system administrators with tooling to evaluate the security of arbitrary rule-based password composition policies.

Other studies such as that by Shay et al. [31] actively recruit users to create passwords under various password composition policies, and attempt to quantify the security of those policies by running password cracking attacks against passwords collected under these policies. This is considered by many to represent the gold standard of password composition policy research, and as such we replicate results from Shay et al. [31] in Section 5.2 to validate our novel methodology.

Regardless of how it is obtained, it is of vital importance that any model designed to evaluate the effectiveness of password composition policies in reducing the vulnerability of human-chosen passwords to guessing attacks is in some way informed by human-generated password data. Password choice varies significantly across different user demographics (age and nationality for example [5]) and by extension across password-protected systems which have user bases comprising different proportions of these demographics. By consequence of this variability, there can be no definitive password composition policy that will lead to ideal security or usability outcomes across all systems—such policies must be designed on a system-by-system basis. Work by Galbally et al. [16] reaffirms this idea—no password strength estimation metric is ideal for all passwords under all conditions. With this in mind, the methodology presented in our work is designed to be attack-independent, and provide a general idea of the security of password composition policies when deployed "in the wild" where the shape of password guessing attacks the system might be subjected to can seldom be known in detail. The only assumption we make about the threat model we face is that the attacker is attempting to guess more common passwords first.

As the weakest passwords are, ostensibly, those that are the most likely to be chosen by users, we can think of the ideal password composition policy as the one that induces the most uniform password distribution on our system. Password policies with poor usability will cause users to converge on fewer easy-to-remember passwords and those with poor security will permit the selection of very weak passwords such as "password" and "123456". This is not a novel argument. Work on adaptive password composition policies [30] supports the view that greater password diversity is key to system security while research into password composition policy optimisation [4] focuses on maximising minimum password entropy—that is, reducing the probability of the most likely password, analogous to increasing password distribution uniformity. Malone and Maher [26] highlight that user-chosen password distributions are non-uniform, and mention that if this were not the case, attacks that rely on attempting to guess common passwords would become less effective.

3 METHODOLOGY

In this section, we present our methodology for rigorous and justifiable password composition policy selection in detail, beginning with raw password data and ending with arbitrary user-specified password composition policies ranked under various assumptions about user behaviour.

3.1 Sourcing Human-Chosen Passwords

With the variability of user password choice in mind [5], our methodology is parametric on an *input set*—some collection of password data that we expect to be representative of the user base we are modelling, given as a password frequency distribution. Input sets can be sourced from any user credential database where the password plaintext is known, but those used within this work include:

- **RockYou**—compromised in plaintext from the *RockYou* online gaming service of the same name around the year 2009 [11]. The password composition policy in place at the time enforced a minimum length of 5 characters with no other requirements [17]. The version we obtained contained 32,603,048 passwords.
- **Yahoo**—compromised in plaintext from the Yahoo Voices online publishing platform around the year 2012 [18]. The password composition policy in place at the time of the breach enforced a minimum length of 6 characters with no other requirements [27]. The version we obtained contained 453,492 passwords.
- **LinkedIn**—compromised from the professional social networking site *LinkedIn* around the year 2012. The true extent of this breach was unknown until 2016 when it was revealed to be much more extensive than was initially made public [6]. Unsalted password hashes in SHA-1 format were compromised, and ≈ 98% of these have subsequently been cracked. It is these cracked passwords that make up the LinkedIn dataset we use in this work. The policy in place at the time of the breach enforced a minimum length of 6 characters [27]. The version we obtained contained 172,428,238 passwords.

3.1.1 Data cleansing. For a dataset to be as representative as possible, each password within it must have been created by a human under a known password composition policy which has a permitted password space that is a superset of that of the password composition policies we wish to model. It is therefore useful to filter these datasets according to the password composition policy they were created under in order to remove any passwords created under old password composition policies or non-password artifacts [22] that might be present within them. In cases where this policy is not known, it is possible to attempt to infer it using a password composition policy inference tool such as *pol-infer* [21]. Each dataset was first filtered according to the password composition policy it is known to have been created under. The small proportion of passwords containing non-ASCII characters were then removed to avoid encoding issues that might arise due to multi-byte characters being stored as multiple characters, artificially inflating password length. Some passwords in the Yahoo dataset (10,654 passwords) appeared to be single sign-on flags for integration with an external service, and were accordingly removed. Likewise, some passwords in the LinkedIn dataset (174,088 passwords) appeared to be hexadecimal data (perhaps due to encoding issues), and were also removed. The sizes of each dataset used in this study after this filtration step are shown in Table 1.

Table 1: A breakdown of the number of passwords filtered from each dataset used in this study.

Dataset	Filtered size	Removed
RockYou [11]	32,506,433	96,615 (0.30%)
Yahoo [18]	434,287	19,205 (4.23%)
LinkedIn [6]	172,235,601	192,637 (0.11%)

3.1.2 Frequencies to probabilities. Following Blocki et al. [3], given a cleansed input set I of N user passwords, we use f_i to denote the frequency of the i^{th} most common password in the set and pwd_i to denote the i^{th} most common password in the set.

The set I induces a probability distribution D over passwords defined as:

$$D(p) = \begin{cases} \frac{f_i}{N} & \text{if } p = pwd_i \\ 0 & \text{otherwise} \end{cases}$$

The probability $D(p)$ is the probability that a random user selects password p. We define the magnitude of the distribution induced by I as the number of passwords in I. That is, $\mathrm{mag}(D) = N$.

3.2 Specifying Password Composition Policies

Our methodology is not tied to any specific representation of password composition policies. Similar to Blocki et al. [4], we use a set-theoretic notation, with $p \in \phi$ indicating that a password p is permitted by a password composition policy ϕ. Later on in Section 4.1, when we describe our encoding of password composition policies in Skeptic, we will demonstrate that this affords us the power to encode password composition policies for arbitrary software, and scaffold code for doing so automatically.

3.2.1 *Policies studied in this work.*

We selected and modelled a selection of password composition policies based on those by Shay et al. [31] and Weir et al. [36], and follow the naming convention used by Shay et al. [31] as follows:

- **basic7, basic8, basic9, basic12, basic14, basic16, basic20**: to comply with policy *basicN*, password must be N characters or greater in length. No other requirements.
- **digit7, digit8, digit9, digit10**: to comply with policy *digitN*, password must be N characters or greater in length, and contain at least one numeric digit.
- **upper7, upper8, upper9, upper10**: to comply with policy *upperN*, password must be N characters or greater in length, and contain at least one uppercase letter.
- **symbol7, symbol8, symbol9, symbol10**: to comply with policy *symbolN*, password must be N characters or greater in length, and contain at least one non-alphanumeric character.
- **2word12, 2word16**: to comply with policy *MwordN*, password must be N characters or greater in length and consist of at least M strings of one or more letters separated by a non-letter sequence.
- **2class12, 2class16, 3class12, 3class16**: to comply with policy *NclassM*, password must be M characters or greater in length and contain at least N of the four character classes (uppercase letters, lowercase letters, digits and symbols).
- **dictionary8**: to comply with policy *dictionaryN* password must be N characters or greater in length. When all non-alphabetic characters are removed the resulting word cannot appear in a dictionary, ignoring case (we used the Openwall "tiny" English wordlist [28]).
- **comp8**: to comply with policy *compN* password must comply with *dictionaryN* and additionally must contain uppercase letters, lowercase letters, digits and symbols. Replicates the NIST comprehensive password composition policy [7].

3.3 Modelling Password Reselection

If a potential user is forbidden from selecting their preferred password by the password composition policy, they must select a different, compliant password or find themselves unable to use the service at all. In this way, a password composition policy induces a change in the probability distribution of passwords on the system.

In this section, we consider the change induced in a probability distribution D by imposing a password composition policy ϕ. In what follows, we write $\mathrm{supp}(D)$ to denote the support of distribution D, that is:

$$\mathrm{supp}(D) = \{\, p \mid D(p) \geq 0 \,\}$$

and we write $\mathrm{supp}_\phi(D)$ to denote the support of D restricted to passwords that comply with ϕ:

$$\mathrm{supp}_\phi(D) = \{\, p \mid p \in \mathrm{supp}(D) \wedge p \in \phi \,\}$$

We assume that $\mathrm{supp}_\phi(D)$ will always be non-empty.

The change induced in D by ϕ can be seen as a redistribution of the probabilities associated with passwords that do not comply with the password composition policy. The sum of the probabilities that need to be redistributed is denoted as $\mathrm{surplus}(D, \phi)$ and defined as:

$$\mathrm{surplus}(D, \phi) = \sum_{\substack{p \in \mathrm{supp}(D) \\ p \notin \phi}} D(p)$$

Figure 1 shows a minimal example of a probability distribution derived from a hypothetical password dataset consisting of 31 user-chosen passwords, of which 5 are unique, labelled P_1 to P_5 with frequencies following the powers of 2. That is to say, the frequency $freq(P_n)$ of password P_n is 2^{5-n} and the probability $D(P_n)$ of password P_n is $\frac{1}{2^n}$.. In this section, we visualise the effect of different reselection modes on this simple example.

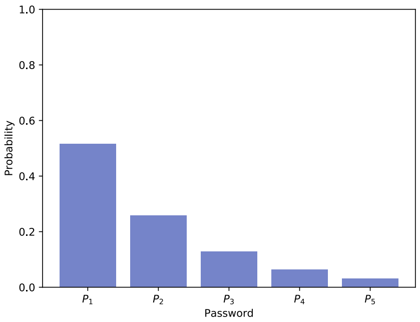


Figure 1: The simple, minimal example of a password probability distribution that we use to visualise different reselection modes in this section. Probability $D(P_n)$ of password P_n is $\frac{1}{2^n}$.

While it would be impossible to accurately predict this reselection process for each individual affected user, we can model certain behaviours that, if exhibited by all users, would give rise to a best, worst, or average-case security outcome. We refer to these as *macrobehaviours*, and examine four of these as part of this work (though our implementation is modular, see Section 4). Given a specific macrobehaviour, the induced distribution obtained from imposing a password composition policy ϕ in a password probability distribution D is denoted as:

$$\mathrm{Reselection}(D, \phi, \mathrm{macrobehaviour})$$

3.3.1 Convergent reselection.

Every user that must reselect a password chooses the most common password that remains permitted (i.e. password choice *converges* on the most common permitted password). This represents a worst-case security outcome; a larger proportion of users now have the same password, which makes the password probability distribution less uniform and the system more vulnerable to a password guessing attack containing this password.

Formally, we define this reselection mode as:

$$\mathsf{Reselection}(D, \phi, \mathsf{convergent})(p) = \begin{cases} D(p) + \mathsf{surplus}(D, \phi) & \text{if } p = \mathsf{max}_\phi(D) \\ D(p) & \text{if } p \neq \mathsf{max}_\phi(D) \text{ and} \\ & \quad p \in \mathsf{supp}_\phi(D) \\ 0 & \text{otherwise} \end{cases}$$

Here, $\mathsf{max}_\phi(D)$ denotes the password with highest probability in D that satisfies the password composition policy ϕ. This can be defined as:

$$\mathsf{choose}(\{\, p \mid p \in \mathsf{supp}_\phi(D) \wedge \\ \forall p' \bullet p' \in \mathsf{supp}_\phi(D) \longrightarrow D(p) \geq D(p') \,\})$$

where choose is non-deterministic choice of one element from the given set (which is non-empty).

Figure 2 shows a simple example of convergent reselection applied to the example distribution shown in Figure 1 when a password composition policy prohibiting passwords P_1 and P_2 is applied. Note that the probability from these prohibited passwords is redistributed to the most common password P_3 in the dataset that remains permitted.

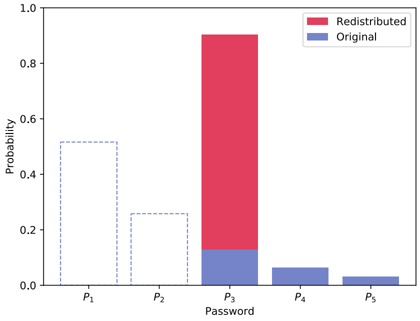


Figure 2: The redistribution of probability in convergent reselection mode under a policy prohibiting P_1 and P_2. Dotted bar outlines show the probability of prohibited passwords, and stacked bars show the redistribution of this probability.

3.3.2 Proportional reselection.

Every user that must reselect a password chooses a password from those remaining in a way proportional to their probabilities. This represents an average-case security outcome, with the most common remaining permitted passwords receiving the largest share of “displaced” users.

Formally, we define this reselection mode as:

$$\mathsf{Reselection}(D, \phi, \mathsf{proportional})(p) = \begin{cases} \dfrac{D(p)}{1 - \mathsf{surplus}(D, \phi)} & \text{if } p \in \mathsf{supp}_\phi(D) \\ 0 & \text{otherwise} \end{cases}$$

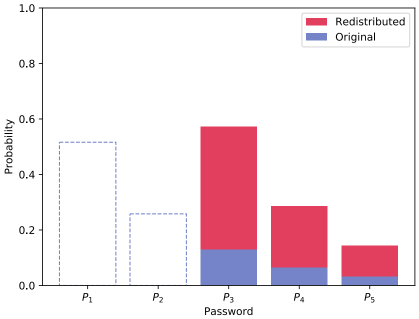


Figure 3: The redistribution of probability in proportional reselection mode under a policy prohibiting P_1 and P_2.

Figure 3 shows a simple example of proportional reselection applied to the example distribution under a policy prohibiting P_1. and P_2 Note that the probability from these prohibited passwords is redistributed amongst remaining permitted passwords proportionally to their probability.

3.3.3 Extraneous reselection.

Every user that must reselect a password chooses a new, unique password outside the set of remaining passwords, as if they had suddenly switched to using a password manager. This represents a best-case security outcome, increasing password probability distribution uniformity to the greatest extent.

Formally, we define this reselection mode as:

$$\mathsf{Reselection}(D, \phi, \mathsf{extraneous})(p) = \begin{cases} D(p) & \text{if } p \in \mathsf{supp}_\phi(D) \\ \dfrac{1}{n} & \text{if } p \in \mathsf{fresh}(S, \phi, D, n) \\ 0 & \text{otherwise} \end{cases}$$

where $n = \mathsf{surplus}(D, \phi) \times \mathsf{mag}(D)$ and $\mathsf{fresh}(S, \phi, D, n)$ is a set of n new and unique passwords built from symbols in the alphabet S that satisfy policy ϕ. Formally, it is a set that satisfies:

$$|\mathsf{fresh}(S, \phi, D, n)| = n$$

and

$$\mathsf{fresh}(S, \phi, D, n) = \{\, p \mid p \in \phi \wedge p \notin \mathsf{supp}(D) \wedge p \in S^* \,\}$$

Figure 4 shows a simple example of proportional reselection applied to the example distribution under a policy prohibiting P_1 and P_2. Note that the probability from these prohibited passwords is redistributed to new, unique passwords P_6-P_{29}.

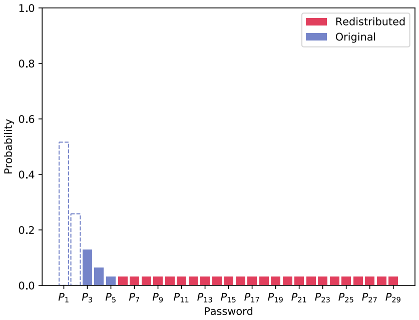


Figure 4: The redistribution of probability in extraneous reselection mode under a policy prohibiting P_1 and P_2.

3.3.4 Null reselection. Every user that must reselect a password simply doesn't, and never creates an account on the system. This is modelled while maintaining the probability distribution by distributing password probability completely evenly amongst all remaining permitted passwords.

Formally, we define this reselection mode as:

$$\text{Reselection}(D, \phi, \text{null})(p) = \begin{cases} D(p) + \dfrac{\text{surplus}(D, \phi)}{|\text{supp}_{\phi}(D)|} & \text{if } p \in \text{supp}_{\phi}(D) \\ 0 & \text{otherwise} \end{cases}$$

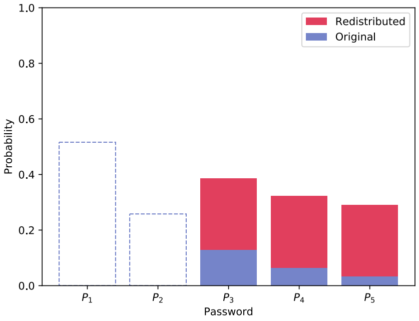


Figure 5: The redistribution of probability in null reselection mode under a policy prohibiting P_1 and P_2.

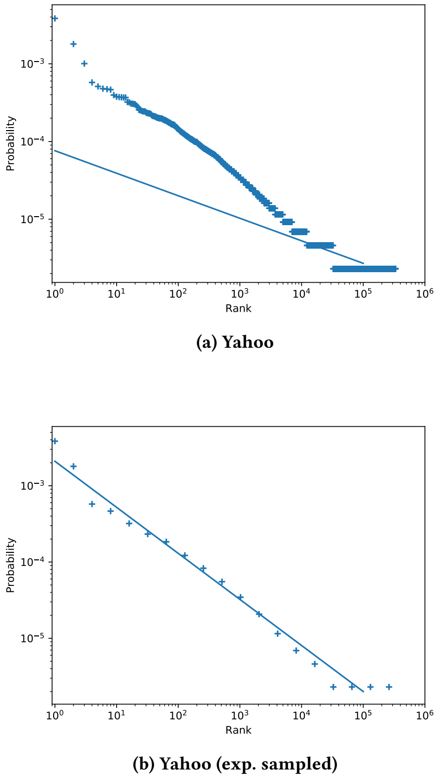
Figure 5 shows a simple example of null reselection applied to the example distribution under a policy prohibiting P_1 and P_2. Note that the probability from these prohibited passwords is redistributed uniformly across remaining permitted passwords.

3.4 Quantifying Security

After transforming our probability distribution according to the policies and macrobehaviours we wish to study, we are now faced with the challenge of quantifying what it means for a distribution of user-chosen passwords to be "secure". To achieve this, we take advantage of the fact that more uniform distributions of user-chosen passwords are more resilient against certain password guessing attacks that rely on guessing common passwords first, due to a smaller proportion of users converging on the same popular passwords. The notion of uniformity as a desirable property of the distribution of user-chosen passwords on a system is not new:

- Previous work by Segreti et al. [30] proposes password composition policies that are *adaptive*—evolving over time with the express aim of increasing password diversity.
- Blocki et al. [4] focus on maximising minimum password entropy in order to optimise password composition policies—analogous to increasing password distribution uniformity.
- Malone and Maher [26] highlight that user-chosen password distributions are non-uniform, and mention that if this were not the case, attacks that rely on attempting to guess common passwords would become less effective.

We approach the problem of measuring the uniformity of password probability distributions by performing least-squares fitting of power-law equation to them of the form $y = a \times x^{\alpha}$. By taking α (the "α-value" of the policy), we can compare the steepness of the fitted curves, with a shallower curve (i.e. a curve with an α-value closer to 0) signifying a more uniform distribution.



(a) Yahoo

(b) Yahoo (exp. sampled)

Figure 6: The rank-probability distribution of passwords in the Yahoo dataset, with and without exponential sampling.

This is not completely straightforward, however. Malone and Maher [26] point out that the tendency for breached password databases to contain a high proportion of passwords with frequencies in the low-single digits causes a least-squares regression line fitted to a graph of password rank against frequency (and therefore probability) to have a slope that is too shallow (see Figure 6a). Logarithmic binning of this data (that is, summing all frequencies between rank 2^n and 2^{n+1} as one data point) removes this bias, and results in a much better fit. We reproduce this result for the Yahoo data set [18] (which we will discuss in detail later) in Figure 6b, but with an important difference—instead of summing the frequencies in each bin, we simply take every 2^{nth} data point and discard those in between; that is to say, we swap logarithmic binning for *exponential sampling*. This similarly corrects our regression line, which now appears to interpolate the data well. Given the *rank* of the probability of a password in the database between 1 and the total number of unique passwords in the database, we can now approximate its actual probability using only the fitted equation, without requiring access to the password data itself. This allows us to justify our choice of password composition policy while avoiding the ethical concerns involved in propagating the password data that informed this choice.

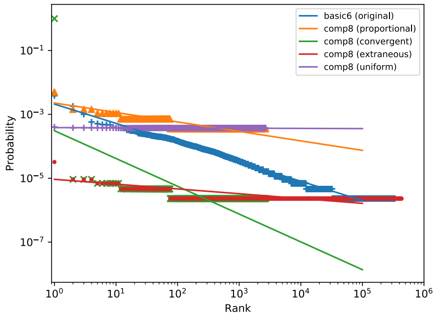


Figure 7: The original password probability distribution of the Yahoo dataset, alongside those induced by the *comp8* policy under each macrobehaviour. Fitted power-law curves are also shown.

Figure 7 shows the rank-probability distribution of the Yahoo dataset used in this study under its original policy (*basic6*) and its transformations under the *comp8* policy assuming each of the macrobehaviours described in Section 3.3. From the figure, it is readily apparent that different assumptions about user password reselection behaviour can lead to drastically different security outcomes for the system. While proportional, extraneous and null reselection behaviours lead to a net increase in uniformity under the *comp8* policy (and therefore presumed guessing attack resistance) convergent behaviour leads to a drastic decrease.

4 THE SKEPTIC TOOLCHAIN

We provide an implementation of the methodology in Section 3 as a toolchain consisting of three pieces of software, designed to be used together sequentially. We name this three-part toolchain Skeptic, which consists of: the metaprogramming tool Authority for encoding password composition policies from within the *Coq* proof assistant; the data processing tool Pyrrho for redistributing password probabilities in the input set according to a password composition policy and user behaviour model; and finally Pacpal, a DSL to assist system administrators in comparing and ranking password composition policies based on output from these tools. We elaborate on each of these in turn in this section.

4.1 Policy Specification: Authority

Password composition policies are enforced on different systems by a diverse range of software, which may accept password policies in different encodings. It is convenient to represent these encodings as tuples containing software configuration parameters. For example, software A may take a tuple $(l \in \mathbb{N}, d \in \mathbb{N})$ where l is minimum password length and d is the minimum number of numeric digits a password may contain; while software B might take tuples $(e \in \mathbb{Q}, w \subset S^*)$ where e is the minimum Shannon entropy of the password, and w is a set of prohibited passwords (a "dictionary check"). If we wish to compare one of each of these tuples, we must first obtain them in a uniform (i.e. normalised) encoding.

To achieve this, we take advantage of the fact that any password composition policy is necessarily a predicate on passwords (i.e. a function with type $Password \rightarrow \mathbb{B}$). With this in mind, we can obtain a uniform representation of password composition policies regardless of the software they were encoded for by devising a function to decode them to a Boolean normal form. For software A for example, we might devise the function in Equation 1 which will transform a password composition policy encoded for this software into a predicate in conjunctive normal form.

$$norm_A(l, d) = \lambda s.length(s) \geq l \wedge digits(s) \geq d \tag{1}$$

Even though software B takes a different configuration tuple, we need only specify the normalisation function in Equation 2 for tuples of this type in order to obtain a password composition policy predicate in the same representation.

$$norm_B(e, w) = \lambda s.shannon(s) \geq e \wedge s \notin w \tag{2}$$

Normalisation functions specified in this way are amenable to formal verification, not only with respect to their correctness (i.e. their conversion of software-specific configuration tuples to predicates) but also desirable properties of the predicates they generate. For instance, we can show that a policy mandating a minimum password length of 16 encoded for software A as configuration tuple $(16, 0)$ and normalised to policy predicate ϕ confers immunity to a guessing attack consisting of passwords in an arbitrary set of guesses G by showing the universal quantification in Equation 3 holds.

$$\phi = norm_A(16, 0) \qquad \forall g \in G. \neg\phi(g) \tag{3}$$

Authority is a metaprogramming utility[1] that enables the interactive modelling of password composition policies for arbitrary software, generating a *Coq* project. From the *Coq* interactive theorem proving environment, it is then possible to both specify and verify the correctness of a normalisation function for transforming password composition policies encoded as software-specific tuples

[1] We make Authority available as open-source software: https://github.com/sr-lab/skeptic-authority-template/

into predicates (see Section 3.2) as well as desirable properties of the password composition policies themselves, such as immunity to certain guessing attacks that malware uses to propagate (see Section 5.5). This command-line utility asks the user a series of questions, guiding them through this process:

(1) They are first asked to specify the name, type and description of each member of the type of software-specific configuration tuple they wish to model.
(2) Then, they may optionally specify an arbitrary number of different password composition policies encoded as tuples of this type by specifying policy names and tuple values.
(3) A ready-to-use *Coq* project is then generated according to the user's specifications. All that remains is for the user to manually specify the normalisation function (see Section 3.2) to convert the password composition policy tuples into predicates.

For a more detailed overview of the operation of Authority, see the flow diagram in Figure 8. Included in the generated *Coq* project are various tools designed to streamline the process of proving desirable properties about the password composition policies encoded using the tool, including a trie implementation for high-performance dictionary checks, a pre-built notion of immunity and a simple `simulate` tactic that can be used to prove properties about password composition policies with respect to smaller guessing attacks by simple simulation.

A central feature of Authority is that is can be used by Pyrrho, the next utility in the Skeptic toolchain, to filter large sets of real-world user password data in order to model changes in the distribution of passwords under different password composition policies and user macrobehaviours. Password composition policies can therefore be modelled from within *Coq*, and used directly for this filtration step. Authority achieves this by making use of the *Coq.io* [9] library for writing IO-enabled programs in *Coq*, and communicating with Pyrrho (which is written in Python for optimal performance) via its standard output stream.

4.2 Password Reselection: Pyrrho

Pyrrho lies at the core of the Skeptic toolchain, a software tool[2] written in Python that handles the transformation of password probability distributions derived from real-world datasets according to password composition policies and assumptions about user behaviour (i.e. the macrobehaviours discussed in Section 3.3). Figure 9 shows an overview of the Skeptic toolchain, and the position of Pyrrho within it, with arrows indicating the direction of data flow between tools.

The utility is parametric on a password probability distribution derived from a real-world leaked password dataset. Password probabilities are then redistributed according to a password composition policy (interpreted by Authority), producing output distributions under each supported macrobehaviour. Its architecture is modular, allowing user-specified macrobehaviours to be plugged in without any modification to the core of the tool. The Pyrrho plugin corresponding to the proportional password reselection macrobehaviour from Section 3.3 is shown in Figure 10. Here, `total` is the

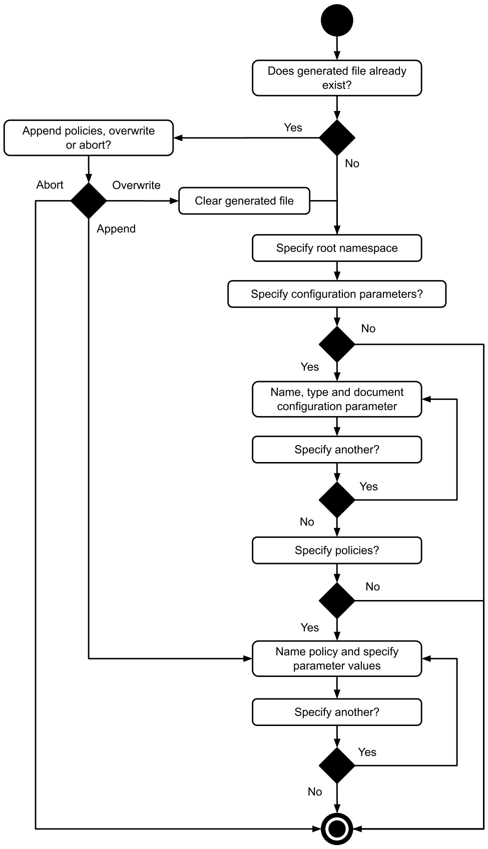

Does generated file already exist?
Yes
No
Append policies, overwrite or abort?
Abort
Overwrite
Append
Clear generated file
Specify root namespace
Specify configuration parameters?
No
Yes
Name, type and document configuration parameter
Specify another?
Yes
No
Specify policies?
No
Yes
Name policy and specify parameter values
Specify another?
Yes
No

Figure 8: A simplified overview of the logical flow of a run of the Authority utility.

sum of all probabilities in the distribution before filtration (which should be ≈ 1), `surplus` is the sum of the probabilities of all filtered passwords, and `df` is the data frame representing the password probability distribution to process.

Pyrrho additionally performs power-law curve fitting to the altered password probability distributions in order to quantify their uniformity (see Section 3.4), storing the resulting equations encoded as JSON files alongside them. It is these JSON files that can be used to compare and rank policies from the Pacpal DSL (see Section 4.3).

While Pyrrho is primarily designed to be used alongside password composition policies encoded in *Coq* using Authority, the inter-process communication involved between the two utilities makes processing large datasets a time-consuming process. For applications where the ability to reason about password composition policies from within *Coq* is less important, Pyrrho also supports *Pure Python Mode*, in which all dataset filtration with respect to a password composition policy is kept within Pyrrho itself. The result is a utility which runs on the order of 2.75 times faster (see Section 5.1), but at the expense of the flexibility of password composition policy encoding and reasoning that comes with using Authority, as *Pure Python Mode* supports only a limited set of password composition policy rules.

[2] We make Pyrrho available as open-source software:
https://github.com/sr-lab/pyrrho

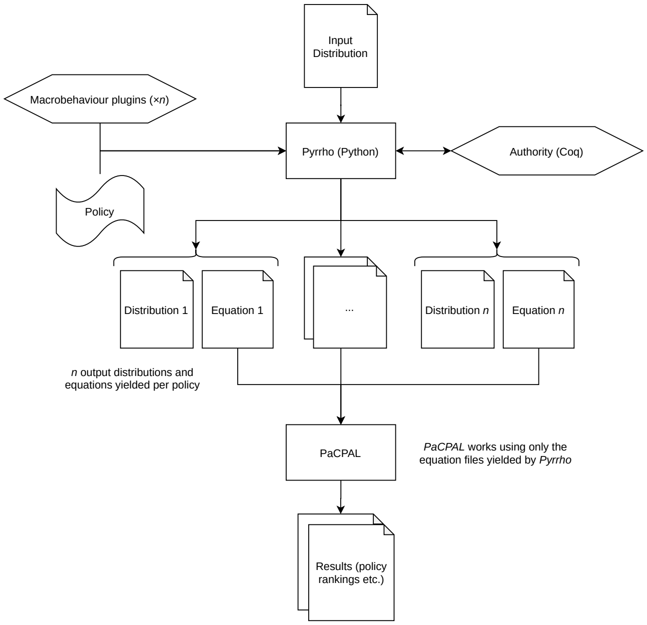

Input Distribution
Macrobehaviour plugins (×n)
Pyrrho (Python)
Authority (Coq)
Policy
Distribution 1
Equation 1
...
Distribution n
Equation n
n output distributions and equations yielded per policy
PaCPAL
PaCPAL works using only the equation files yielded by Pyrrho
Results (policy rankings etc.)

Figure 9: An overview of the function of Skeptic. Arrows indicate the direction of data flow.

```python
def reselect (total, surplus, df):
    divisor = total - surplus
    df['probability'] /= divisor
    return df
```

Figure 10: The proportional password reselection macrobehaviour from Section 3.3 encoded in Python as a plugin for Pyrrho.

4.3 Result Extraction: Pacpal

While the data produced by Pyrrho is ostensibly all we need to be able to assess the relative security of password composition policies under our assumptions, the nuance of this data is of comparatively little interest to professionals working in an applied setting (system administrators, for example).

Users such as this are likely to be far more interested in choosing the most secure password composition policy for their use-case than in the data itself. Pacpal[3] is an assertion language permitting power-law equations generated by Pyrrho to be loaded, named, grouped, compared and ranked, and is designed to assist end-users in putting Skeptic to work practically in their organisations, leveraging the well-documented usability benefit seen with domain-specific languages when compared to their general-purpose counterparts [2]. An example piece of Pacpal code is shown in Figure 11 in which three fitted power-law equation files produced by Pyrrho are loaded, bound to names, added to a group and ranked. The ranking will then be displayed to the user. Also present is a `better` assertion which will display an error to the user in the case that this relationship does not hold. We employ Pacpal to produce the rankings of all 28 password policies used in this study in Section 5.4.

```
# Load three equations produced by Pyrrho.
load linkedin-basic16-proportional.json as li_b16
load linkedin-2word16-proportional.json as li_2w16
load linkedin-3class12-proportional.json as li_3c12

# Assert that one policy is better than another.
assert li_2w16 better li_b16

# Build group to rank.
group linkedin_ranking
add li_b16 to linkedin_ranking as basic16
add li_2w16 to linkedin_ranking as 2word16
add li_3c12 to linkedin_ranking as 3class12

# Print group in ranked order (worst to best):
rank linkedin_ranking
```

Figure 11: A piece of example Pacpal code, demonstrating ranking of policies based on fitted power-law equations.

5 EVALUATION

In this section, we demonstrate the validity of our approach by replicating results from previous literature across different evaluation methodologies. Specifically, we use the Skeptic toolkit to replicate results from the study by Shay et al. [31] that uses real participants recruited via Amazon Mechanical Turk (see Section 5.2), and the study by Weir et al. [36] that draws on large leaked password datasets (see Section 5.3). In Section 5.5, we demonstrate the advantages of the Authority *Coq* metaprogramming utility (see Section 4.1) by proving that certain policies confer immunity to password guessing attacks by some common botnet worms from within the proof assistant itself.

5.1 Experimental Setup

The password probability distribution processing (via Pyrrho) for this experiment was conducted on a cluster of 14 cloud-based virtual machines, each with 6 Intel® Xeon® CPUs at 1.80GHz, 16GB of RAM and 320GB of hard disk space running 64-bit Ubuntu 18.04.3 (LTS). Times taken by Pyrrho to process each dataset studied in this work under each policy and macrobehaviour we investigate are shown in Table 2.

5.2 Replication of Results: Shay et al.

Shay et al. [31] ranked the effectiveness of 8 different password composition policies under a password guessing attack at two different magnitudes—10^6 guesses and 10^{14} guesses. These two thresholds are suggested by Florêncio et al. [15] as being representative of the cutoff points of contemporary online (i.e. against a live service) and offline (i.e. against a compromised password hash) guessing attacks respectively. Passwords were chosen by humans under each policy

[3] We make Pacpal available as open-source software: https://github.com/sr-lab/skeptic-lang

Table 2: Time taken for Pyrrho to process probability distributions for each of the datasets, policies and macrobehaviours investigated.

Dataset	Time (s)	Uniq. passwords	Time/password
Yahoo	17,817	337,168	0.0528
Yahoo*	6,466	337,168	0.0192
RockYou*	339,708	14,308,965	0.0237
LinkedIn*	1,741,996	60,489,959	0.0288

* Computed in *Pyrrho's* pure Python mode for reasons of performance.

using Amazon Mechanical Turk and the attack was multimodal using both a trained, targeted probabilistic context-free grammar (PCFG) [22, 37] and the *Password Guessability Service* (PGS) [33]. Table 3 contains an overview of these results.

Table 3: The results obtained by Shay et al. [31] for passwords collected under 8 different password composition policies at both attack magnitudes.

	10^6 guesses		10^{14} guesses	
Policy	Cracked (%)	Rank	Cracked (%)	Rank
comp8	2.2	3	50.1	7
basic12	9.1	8	52	8
basic16	7.9	7	29.7	4
basic20	5.6	6	16.4	2
2word12	3.4	5	46.6	6
2word16	1.1	1	22.9	3
3class12	3.2	4	36.8	5
3class16	1.2	2	13.8	1

We attempted to replicate these results using the Skeptic toolkit. For each of our 3 datasets, and each of the 4 studied macrobehaviours, we redistributed probability according to each policy in Table 3. We then obtained the α values yielded by fitting power-law curves to the resulting distributions using the methodology described in Section 3.4. In order to quantify how closely our results reflect the rankings from Shay et al. [31] we plotted the percentage of passwords cracked under each policy in Shay et al. [31] against the α-values we obtained using our methodology and calculated the Pearson correlation coefficient ρ. A value closer to -1 indicates that more uniform distributions (i.e. a less negative α-value) are more strongly correlated with a lower percentage of cracked passwords according to Shay et al. [31], while a value closer to 1 indicates the opposite. A value of 0 indicates no correlation. The complete set of correlation coefficients and their mean values across datasets $\bar{\rho}$ can be found in Table 4, while an example visualisation using the LinkedIn dataset only is shown in Figure 12. Complete results are shown in the Appendix (Table 11).

From Table 4, it is apparent that α-values for proportional, extraneous and null macrobehaviours tend to correlate well with the empirical results from Shay et al. [31]. Using thresholds proposed by Evans [12], correlation strengths range from *moderate* ($0.40 \leq |\rho| \leq 0.59$) to *very strong* ($0.80 \leq |\rho| \leq 1.0$) for each of these macrobehaviours across all 3 datasets, with an average correlation strength of *strong* ($0.60 \leq |\rho| \leq 0.79$). By contrast, the convergent macrobehaviour tends to show a correlation in the opposite direction, with less uniform distributions being associated with lower percentages of cracked passwords. This suggests the convergent macrobehaviour is a poor model of how users actually reselect passwords in response to password composition policies.

Table 4: Pearson correlation coefficients of percentage of passwords cracked under different polices by Shay et al. [31] at 10^{14} guesses against α-values yielded by Skeptic.

Mode	Yahoo	RockYou	LinkedIn*	$\bar{\rho}$
Proportional	-0.661	-0.591	-0.929	-0.727
Convergent	0.882	-0.069	0.615	0.476
Extraneous	-0.722	-0.689	-0.952	-0.788
Null	-0.550	-0.565	-0.884	-0.666

* Visualised in Figure 12.

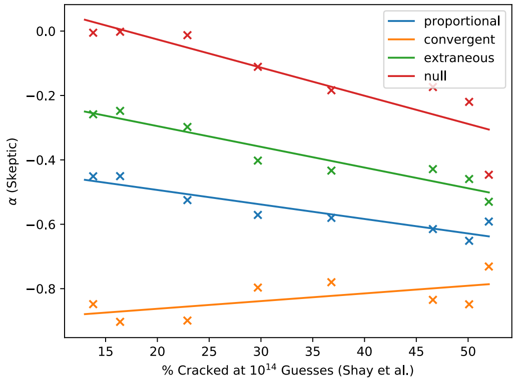


Figure 12: Percentage of passwords cracked by Shay et al. [31] at 10^{14} guesses against α-values yielded by Skeptic for the LinkedIn dataset in each reselection mode.

We found α-values yielded by Skeptic to correlate slightly less closely with the percentage of passwords cracked by the smaller online-range guessing attack from Shay et al. [31] (see Table 5 and Figure 13). We imagine that this is due to the success of smaller guessing attacks being more dependent on the dataset they are performed against. It is also possible that the multimodal attack employed by Shay et al. [31] is causing guessing attacks at lower magnitudes to be more effective against passwords created under different password composition policies than at higher magnitudes.

Table 5: Pearson correlation coefficients of percentage of passwords cracked under different polices by Shay et al. [31] at 10^6 guesses against α-values yielded by Skeptic.

Mode	Yahoo	RockYou	LinkedIn*	$\bar{\rho}$
Proportional	-0.866	-0.676	-0.149	-0.564
Convergent	0.217	-0.181	0.615	0.217
Extraneous	-0.830	-0.808	-0.462	-0.700
Null	-0.684	-0.797	-0.558	-0.680

* Visualised in Figure 13.

The observation that the proportional, null and extraneous macrobehaviours offer a more accurate picture of user password reselection than convergent reselection is encouraging, because each

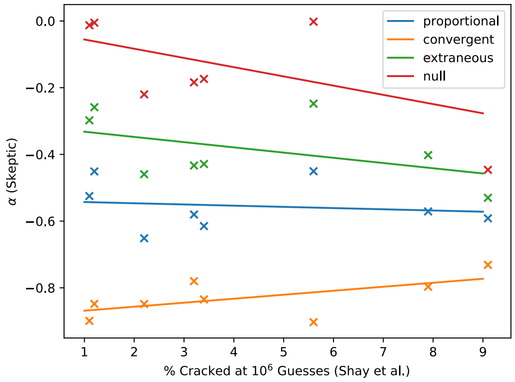


Figure 13: Percentage of passwords cracked by Shay et al. [31] at 10^6 guesses against α-values yielded by Skeptic for the LinkedIn dataset in each reselection mode.

of these represents a net increase (rather than decrease) in the uniformity of the password distribution on the system. This leads us to the conclusion that implementation of stricter password composition policies does, in general, lead to an increase in the resistance of a system to password guessing attacks. Noteworthy, however, are the outlying ρ values for the convergent macrobehaviour on the RockYou dataset (see Tables 4 and 5), which seem to indicate that user password reselection behaviour for this dataset more closely resembles the convergent macrobehaviour. This is possibly due to the age of this dataset in comparison to the others (2009 vs. 2012) and consequently less secure password reselection behaviours by users of that system. This may be demographics and use-case-related, with RockYou being an online gaming service that may have had a higher proportion of younger users less adept at picking secure passwords, or users who place comparatively little value on online gaming accounts compared to those tied directly to their professional or social lives (e.g. the LinkedIn professional social networking site or Yahoo Voices online publishing platform).

Findings. Overall, Skeptic produces α-values, and therefore password composition policy rankings, that are strongly correlated with the results obtained by Shay et al. [31] from real human users recruited to create passwords under various password composition policies. This is particularly true when attack magnitude is greater (e.g. offline attacks) as opposed to smaller attacks in the online range which are more sensitive to the specific password distribution they are conducted against. Because Skeptic takes password distribution uniformity as a measure of security, and thus is attack-independent, this is to be expected. This uniformity-based methodology employed by Skeptic is an accurate measure of general resistance to password guessing attacks, but a considerably poorer measure of resistance to specific, targeted attacks tailored with a specific password distribution in mind.

5.3 Replication of Results: Weir et al.

We next turn our attention to a study by Weir et al. [36] which draws on leaked password datasets in order to attempt to determine password composition policy effectiveness, rather than collecting passwords from humans themselves under those policies.

Table 6: An approximation of the results obtained by Weir et al. [36] for passwords obtained under 12 different password composition policies by filtering their target dataset.

	5×10^4 guesses	
Policy	Cracked (%)	Rank
basic7	26.06	12
basic8	23.16	11
basic9	18.98	10
basic10	13.85	8
upper7	13.89	9
upper8	10.71	7
upper9	7.71	6
upper10	5.72	4
symbol7	6.92	5
symbol8	5.57	3
symbol9	4.76	2
symbol10	3.28	1

This work, among other results, presents the percentage of passwords cracked at 50, 000 guesses under 4 different password length thresholds (7, 8, 9 and 10) and 3 different character requirements (none, at least one uppercase and at least one symbol). Both the target passwords and the attack were drawn from separate subsets of the same RockYou dataset [11] we make use of in this work. We present an approximation of results from [36] in Table 6, obtained using a plot digitiser[4] from the visualisations in the work.

Table 7: Pearson correlation coefficients of password policy ranks from [36] at 5×10^4 guesses against α-values yielded by Skeptic.

Mode	Yahoo	RockYou	LinkedIn	Mean
Proportional	-0.884	-0.916	-0.885	-0.895
Convergent	0.686	-0.657	0.234	0.089
Extraneous	-0.955	-0.951	-0.969	-0.958
Null	-0.953	-0.945	-0.967	-0.955

Under these policies, Skeptic produces α-values that correlate very strongly with the percentage of passwords guessed by Weir et al. [36] in proportional, extraneous, and null reselection modes (see Table 7). The α-values for the LinkedIn dataset under each policy and macrobehaviour studied are plotted against percentages of passwords cracked by Weir et. al [36] in Figure 14.

In convergent reselection mode, Skeptic is much less accurate for the Yahoo and LinkedIn datasets, but retains a strong correlation for the RockYou set. We speculate that this is for the same dataset-specific reasons as presented in Section 5.2 but more pronounced due to the use of the same dataset in both that work, and this one.

Findings. Skeptic produces α-values and policy rankings that are very strongly correlated with results obtained by Weir et al. [36] from large sets of revealed password data.

[4] We used *WebPlotDigitizer*: https://github.com/ankitrohatgi/WebPlotDigitizer

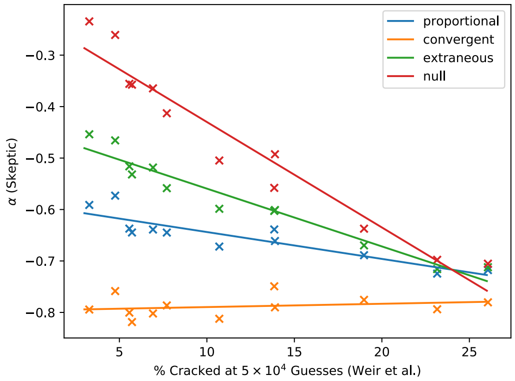


Figure 14: Percentage of passwords cracked in Weir et al. [36] at 5×10^4 guesses against α-values yielded by SKEPTIC for the LinkedIn dataset in each reselection mode.

5.4 Policy Ranking

If we wish to make an informed choice of password composition policy, one way we might go about this is to rank our candidates in order from most to least secure and use the resulting ranking to make our decision. Output from PYRRHO (see Section 4.2) enables us to do this already if we manually extract α-values from each equation file produced and perform additional processing in, for example, spreadsheet software. This introduces a high potential for human error, however, and requires considerable additional data processing work that can be readily automated using the PACPAL DSL (see Figure 11).

Rankings obtained using PACPAL are shown in Table 8. Crucially, we do not require any access to the password data itself to produce these rankings, and thus we avoid the ethical issues involved in propagating user password data while retaining our ability to justify and reproduce these rankings as-needed. We make the PACPAL scripts and equation files necessary to reproduce these results freely available[5].

Findings. We demonstrate that it is possible to use the SKEPTIC toolchain to inform password composition policy choice, and that using the PACPAL DSL this can be done without any additional manual data processing step.

5.5 Policy Immunity

In this section, we demonstrate the utility of encoding password composition policies in the *Coq* proof assistant using AUTHORITY (see Section 4.1) by formally verifying the immunity or vulnerability of 14 password composition policies to the password guessing attacks utilised by the *Mirai* and *Conficker* botnet worms. We achieve this by encoding the notion of vulnerability or immunity to concrete dictionaries of password guesses in Coq and devising a simple `simulate` tactic to prove, by dynamic simulation, assertions that a password composition policy either does or doesn't confer immunity to a guessing attack (see Figure 15).

Table 8: All 28 policies investigated in this work, ranked according to their α-values given by SKEPTIC in proportional reselection mode for each of the 3 datasets studied. Policy ranking performed by PACPAL.

Policy	Yahoo	RockYou	LinkedIn	Average
3class16	1	1	2	1.33
basic20	3	5	1	3
2word16	2	4	5	3.67
2class16	7	3	3	4.33
3class12	4	2	8	4.67
symbol10	9	8	9	8.67
2word12	8	7	11	8.67
symbol9	5	15	7	9
2class12	15	6	12	11
basic14	18	12	4	11.33
comp8	6	9	19	11.33
basic16	19	13	6	12.67
upper9	11	10	18	13
upper10	12	11	17	13.33
basic12	20	14	10	14.67
symbol8	14	18	14	15.33
upper7	10	17	20	15.67
symbol7	16	16	16	16
digit10	17	20	13	16.67
upper8	13	19	22	18
basic10	21	21	15	19
digit9	22	23	21	22
digit7	24	22	24	23.33
digit8	25	24	23	24
basic9	23	26	25	24.67
dictionary8	26	25	26	25.67
basic7	27	28	27	27.33
basic8	28	27	28	27.67

```
(* The `basic14` policy is immune to Mirai. *)
Example basic14_mirai_immune :
  immune "basic14" mirai_dict.
Proof.
  simulate.
Qed.
```

Figure 15: Examples of a proof in *Coq*, showing that the policy named `basic14` renders a system immune to a guessing attack by the *Mirai* malware.

5.5.1 *Mirai.*

Mirai is a piece of malware that targets network-enabled devices running Linux, recruiting them into a botnet that has been used in several high-profile and extremely disruptive distributed denial-of-service (DDoS) attacks to date [23]. In order to propagate, *Mirai* scans IP address ranges for devices with Telnet enabled. Upon locating a potentially vulnerable device, the malware will try a dictionary of 62 username/password combinations (containing 46 unique passwords) containing the factory defaults of a number of common internet-of-things (IoT) devices including CCTV cameras, home routers, and network-enabled printers [1].

Using *Coq*, at the level of the AUTHORITY, we modelled the attack used by *Mirai* to gain access to a device—a dictionary attack consisting of 46 specific guesses. From here, we were able to determine for a selection of the password composition policies by Shay et al. [31] whether or not they render a device immune to *Mirai* when enforced by prohibiting the creation of any vulnerable password.

[5] Access these here: https://github.com/sr-lab/skeptic-example-results

We are confident that these results (see Table 9) would be useful to any company producing Linux-based network-enabled devices. By configuring their devices with a password policy immune to compromise by *Mirai* (such as basic16) before shipping, they are granted assurance that their product cannot be configured to become vulnerable.

Table 9: Whether or not each password composition policy provides immunity to the dictionary attack used by the *Mirai* worm, as verified from within Coq.

Immune	basic14, basic16, basic20, 2class16, 2word16, 3class16, comp8
Vulnerable	basic7, basic8, basic9, basic12, 2class12, 2word12, 3class12

5.5.2 Conficker. Another botnet worm, *Conficker* [32], which first emerged in 2008, remains a considerable threat even today through its use of several different propagation vectors to spread. One of these is a dictionary attack on password-protected administrative shares on Windows systems, which if successful allows the worm to write itself to disk on the remote machine and infect it. The dictionary used by *Conficker* for this purpose is, again, quite small containing only 182 passwords (including the empty password). By encoding our password composition policies from within Coq, we can ascertain whether each password policy from Shay et al. [31] confers immunity against this attack as we did for Mirai. The results of this analysis are shown in Table 10.

Table 10: Whether or not each password composition policy provides immunity to the dictionary attack used by the *Conficker* worm, as verified from within Coq.

Immune	basic14, basic16, basic20, 2class12, 2class16, 2word12, 2word16, 3class12, 3class16, comp8
Vulnerable	basic7, basic8, basic9, basic12

Interestingly, if any of the policies analysed here are immune to *Mirai*, they are also immune to *Conficker* (i.e. the set of policies here that confer immunity to *Mirai* are a subset of those that confer immunity to *Conficker*). We anticipate that researchers will be able to use Skeptic like this to discover policies immune to attack from a wide range of malware.

6 CONCLUSION

In this work, we have demonstrated a new methodology for automatically, rigorously and justifiably selecting the most appropriate choice of password composition policy from a list of candidates. We achieve this by using a user behaviour model and password composition policy to induce a change in password probability distributions derived from large leaked password databases. We then take the uniformity of these distributions as a proxy for their security, demonstrating the validity of this approach by using it to closely reproduce results from two previous studies, one which collected passwords from users under specific password composition policies [31] and one which made use of large breached password datasets [36]. We find that our approach has the advantage of being attack-independent and broadly applicable, with its only assumption being that the attacker attempts to guess more common passwords first, but also that this comes at the expense of the ability to reason accurately about more attacks specifically tailored to target a particular system.

We have also described and presented Skeptic, an implementation of this methodology as a software toolchain consisting of: Authority, a metaprogramming utility for encoding policies in arbitrary representations; Pyrrho a user behaviour model to redistribute probability according to these policies under different assumptions about user password reselection behaviour; and finally Pacpal, a straightforward DSL to make the results of this process accessible to professionals working in the field. In addition, we have used this tool to obtain new results, including: a ranking of all 28 password composition policies studied in this work according to their expected effectiveness at mitigating password guessing attacks, under various assumptions about user password reselection behaviour; a demonstration that under some user behaviour models, certain password composition policies can have a negative effect on password security; and formal verification of the immunity of some password composition policies to the password guessing attacks employed by the *Mirai* and *Conficker* malware.

6.1 Future Work

We are excited about the future of this project, with the design of machine learning-based user behaviour models for password reselection representing a particularly promising potential future research direction. We also plan to expand the capabilities of Pacpal to increase its utility, and explore the possibility of employing the power-law equations fitted by Pyrrho in conjunction with existing password strength estimation algorithms to estimate the success probability of concrete password guessing attacks given as lists of strings.

We are also interested in devising tools and techniques to allow the synthesis of formally verified password composition policy enforcement software such as that by Ferreira et al. [13] from models of password guessing attacks, informed by policy rankings produced by Skeptic. Attack-defence trees in particular [25] appear promising as an intuitive formal representation of password guessing attacks and their mitigation measures from which password composition policies might be synthesised. We have taken some steps towards producing a user-friendly software interface for non-expert users to interact with Skeptic and its satellite tooling with the Passlab project [20], and we believe with further implementation work we will be able to realise a fully-fledged graphical tool for defensive password security.

Acknowledgements. The authors would like to thank the anonymous reviewers, whose expertise and insight was invaluable in shaping it.

REFERENCES

[1] Manos Antonakakis, Tim April, Michael Bailey, Matt Bernhard, Elie Bursztein, Jaime Cochran, Zakir Durumeric, J. Alex Halderman, Luca Invernizzi, Michalis Kallitsis, Deepak Kumar, Chaz Lever, Zane Ma, Joshua Mason, Damian Menscher, Chad Seaman, Nick Sullivan, Kurt Thomas, and Yi Zhou. 2017. Understanding the Mirai Botnet. In *Proceedings of the 26th USENIX Security Symposium.*

[2] A. Bariic, V. Amaral, and M. Goulão. 2012. Usability Evaluation of Domain-Specific Languages. In *2012 Eighth International Conference on the Quality of*

Information and Communications Technology. 342–347. https://doi.org/10.1109/QUATIC.2012.63

[3] Jeremiah Blocki, Benjamin Harsha, and Samson Zhou. 2018. On the economics of offline password cracking. In *2018 IEEE Symposium on Security and Privacy (SP)*. IEEE, 853–871.

[4] Jeremiah Blocki, Saranga Komanduri, Ariel Procaccia, and Or Sheffet. 2013. Optimizing Password Composition Policies. In *Proceedings of the Fourteenth ACM Conference on Electronic Commerce (EC '13)*. ACM, New York, NY, USA, 105–122. https://doi.org/10.1145/2492002.2482552

[5] J. Bonneau. 2012. The Science of Guessing: Analyzing an Anonymized Corpus of 70 Million Passwords. In *2012 IEEE Symposium on Security and Privacy*. 538–552. https://doi.org/10.1109/SP.2012.49

[6] Matt Burgess. 2016. Check if your LinkedIn account was hacked | WIRED UK. https://www.wired.co.uk/article/linkedin-data-breach-find-out-included. (Accessed on 07/26/2019).

[7] William E. Burr, Donna F. Dodson, Elaine M. Newton, Ray A. Perlner, W. Timothy Polk, Sarbari Gupta, and Emad A. Nabbus. 2013. Electronic Authentication Guideline. http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-63-2.pdf. (Accessed on 06/05/2018).

[8] William E. Burr, Donna F. Dodson, and W. Timothy Polk. 2006. Electronic Authentication Guideline. https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-63ver1.0.2.pdf. (Accessed on 06/05/2018).

[9] Guillaume Claret. 2015. Coq.io. http://coq.io/

[10] Nik Cubrilovic. 2009. RockYou hack: From bad to worse. https://techcrunch.com/2009/12/14/rockyou-hack-security-myspace-facebook-passwords. (Accessed on 18/03/2018).

[11] Nik Cubrilovich. 2009. RockYou Hack: From Bad To Worse | TechCrunch. https://techcrunch.com/2009/12/14/rockyou-hack-security-myspace-facebook-passwords/. (Accessed on 04/10/2019).

[12] James D Evans. 1996. *hStraightforward statistics for the behavioral sciences.*

[13] João F. Ferreira, Saul A. Johnson, Alexandra Mendes, and Phillip J. Brooke. 2017. Certified Password Quality - A Case Study Using Coq and Linux Pluggable Authentication Modules. In *Integrated Formal Methods - 13th International Conference, IFM 2017, Turin, Italy, September 20-22, 2017, Proceedings*. 407–421. https://doi.org/10.1007/978-3-319-66845-1_27

[14] Dinei Florêncio and Cormac Herley. 2010. Where Do Security Policies Come from?. In *Proceedings of the Sixth Symposium on Usable Privacy and Security (SOUPS '10)*. ACM, New York, NY, USA, Article 10, 14 pages. https://doi.org/10.1145/1837110.1837124

[15] Dinei Florêncio, Cormac Herley, and Paul C. van Oorschot. 2014. Password Portfolios and the Finite-Effort User: Sustainably Managing Large Numbers of Accounts. In *23rd USENIX Security Symposium (USENIX Security 14)*. USENIX Association, San Diego, CA, 575–590.

[16] J. Galbally, I. Coisel, and I. Sanchez. 2017. A New Multimodal Approach for Password Strength Estimation—Part I: Theory and Algorithms. *IEEE Transactions on Information Forensics and Security* 12, 12 (2017), 2829–2844. https://doi.org/10.1109/TIFS.2016.2636092

[17] Maximilian Golla and Markus Dürmuth. 2018. On the Accuracy of Password Strength Meters. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security (CCS '18)*. ACM, New York, NY, USA, 1567–1582. https://doi.org/10.1145/3243734.3243769

[18] Doug Gross. 2012. Yahoo hacked, 450,000 passwords posted online - CNN. https://edition.cnn.com/2012/07/12/tech/web/yahoo-users-hacked. (Accessed on 04/10/2019).

[19] Troy Hunt. 2018. Have I been pwned? Check if your email has been compromised in a data breach. https://haveibeenpwned.com/. (Accessed on 28/02/2018).

[20] Saul Johnson. 2020. Passlab: A Password Security Tool for the Blue Team. *arXiv preprint arXiv:2003.07208* (2020). [Presented at the Doctoral Symposium at Formal Methods Europe 2019].

[21] Saul Johnson, João F. Ferreira, Alexandra Mendes, and Julien Cordry. 2019. Lost in Disclosure: On The Inference of Password Composition Policies. In *2019 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*.

[22] Patrick Gage Kelley, Saranga Komanduri, Michelle L. Mazurek, Richard Shay, Timothy Vidas, Lujo Bauer, Nicolas Christin, Lorrie Faith Cranor, and Julio Lopez. 2012. Guess Again (and Again and Again): Measuring Password Strength by Simulating Password-Cracking Algorithms. In *Proceedings of the 2012 IEEE Symposium on Security and Privacy (SP '12)*. IEEE Computer Society, Washington, DC, USA, 523–537. https://doi.org/10.1109/SP.2012.38

[23] C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas. 2017. DDoS in the IoT: Mirai and Other Botnets. *Computer* 50, 7 (2017), 80–84. https://doi.org/10.1109/MC.2017.201

[24] Saranga Komanduri, Richard Shay, Patrick Gage Kelley, Michelle L. Mazurek, Lujo Bauer, Nicolas Christin, Lorrie Faith Cranor, and Serge Egelman. 2011. Of Passwords and People: Measuring the Effect of Password-composition Policies. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '11)*. ACM, New York, NY, USA, 2595–2604. https://doi.org/10.1145/1978942.1979321

[25] Barbara Kordy, Sjouke Mauw, Saša Radomirović, and Patrick Schweitzer. 2011. Foundations of Attack–Defense Trees. In *Formal Aspects of Security and Trust*, Pierpaolo Degano, Sandro Etalle, and Joshua Guttman (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 80–95.

[26] David Malone and Kevin Maher. 2012. Investigating the Distribution of Password Choices. In *Proceedings of the 21st International Conference on World Wide Web (WWW '12)*. ACM, New York, NY, USA, 301–310. https://doi.org/10.1145/2187836.2187878

[27] Peter Mayer, Jan Kirchner, and Melanie Volkamer. 2017. 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)*. USENIX Association, Santa Clara, CA, 13–28.

[28] Openwall Project. 2011. Wordlists and common passwords for password recovery. http://www.openwall.com/passwords/wordlists/. (Accessed on 09/01/2018).

[29] Jack Schofield. 2019. I got a phishing email that tried to blackmail me—what should I do? https://www.theguardian.com/technology/askjack/2019/jan/17/phishing-email-blackmail-sextortion-webcam. Accessed: 2019-08-20.

[30] Sean M. Segreti, William Melicher, Saranga Komanduri, Darya Melicher, Richard Shay, Blase Ur, Lujo Bauer, Nicolas Christin, Lorrie Faith Cranor, and Michelle L. Mazurek. 2017. Diversify to Survive: Making Passwords Stronger with Adaptive Policies. In *Thirteenth Symposium on Usable Privacy and Security (SOUPS 2017)*. USENIX Association, Santa Clara, CA, 1–12.

[31] Richard Shay, Saranga Komanduri, Adam L. Durity, Phillip (Seyoung) Huh, Michelle L. Mazurek, Sean M. Segreti, Blase Ur, Lujo Bauer, Nicolas Christin, and Lorrie Faith Cranor. 2016. Designing Password Policies for Strength and Usability. *ACM Trans. Inf. Syst. Secur.* 18, 4, Article 13 (May 2016), 34 pages. https://doi.org/10.1145/2891411

[32] S. Shin, G. Gu, N. Reddy, and C. P. Lee. 2012. A Large-Scale Empirical Study of Conficker. *IEEE Transactions on Information Forensics and Security* 7, 2 (April 2012), 676–690. https://doi.org/10.1109/TIFS.2011.2173486

[33] Blase Ur, Sean M. Segreti, Lujo Bauer, Nicolas Christin, Lorrie Faith Cranor, Saranga Komanduri, Darya Kurilova, Michelle L. Mazurek, William Melicher, and Richard Shay. 2015. Measuring Real-World Accuracies and Biases in Modeling Password Guessability. In *24th USENIX Security Symposium (USENIX Security 15)*. USENIX Association, Washington, D.C., 463–481.

[34] Eric R. Verheul. 2006. Selecting Secure Passwords. In *Topics in Cryptology – CT-RSA 2007*, Masayuki Abe (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 49–66.

[35] D. Wang, H. Cheng, P. Wang, X. Huang, and G. Jian. 2017. Zipf's Law in Passwords. *IEEE Transactions on Information Forensics and Security* 12, 11 (Nov 2017), 2776–2791. https://doi.org/10.1109/TIFS.2017.2721359

[36] Matt Weir, Sudhir Aggarwal, Michael Collins, and Henry Stern. 2010. Testing Metrics for Password Creation Policies by Attacking Large Sets of Revealed Passwords. In *Proceedings of the 17th ACM Conference on Computer and Communications Security (CCS '10)*. ACM, New York, NY, USA, 162–175. https://doi.org/10.1145/1866307.1866327

[37] M. Weir, S. Aggarwal, B. d. Medeiros, and B. Glodek. 2009. Password Cracking Using Probabilistic Context-Free Grammars. In *2009 30th IEEE Symposium on Security and Privacy*. 391–405. https://doi.org/10.1109/SP.2009.8

[38] Daniel Lowe Wheeler. 2016. zxcvbn: Low-Budget Password Strength Estimation. In *25th USENIX Security Symposium (USENIX Security 16)*. USENIX Association, Austin, TX, 157–173.

[39] L. Xu, C. Ge, W. Qiu, Z. Huang, Z. Gong, J. Guo, and H. Lian. 2017. Password Guessing Based on LSTM Recurrent Neural Networks. In *2017 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC)*, Vol. 1. 785–788. https://doi.org/10.1109/CSE-EUC.2017.155

Table 11: Appendix. A complete set of policy α-values rankings for policies evaluated in [31] under each different macrobehaviour studied.

		Policy	Yahoo				RockYou				LinkedIn			
			Shay	Skeptic	α	Distance	Shay	Skeptic	α	Distance	Shay	Skeptic	α	Distance
Reselection modes	Null	3class16	1	1	-0.00015790845	0	1	1	-0.00480967797	0	1	2	-0.00511970014	1
		basic20	2	2	-0.00017481256	0	2	2	-0.00773612979	0	2	1	-0.00206544273	1
		2word16	3	3	-0.00034446767	0	3	3	-0.01310526071	0	3	3	-0.01271757597	0
		basic16	4	6	-0.01237917795	2	4	7	-0.11203436164	3	4	4	-0.11099256297	0
		3class12	5	5	-0.00946485322	0	5	5	-0.01818160822	0	5	6	-0.18384198515	1
		2word12	6	7	-0.01360245343	1	6	6	-0.07942172914	0	6	5	-0.17379245775	1
		comp8	7	4	-0.00619759948	3	7	4	-0.01573345733	3	7	7	-0.21988288974	0
		basic12	8	8	-0.16874098618	0	8	8	-0.32090018785	0	8	8	-0.44625701959	0
	Proportional	3class16	1	1	-0.15000000183	0	1	1	-0.32803183792	0	1	2	-0.45101422402	1
		basic20	2	3	-0.22731830237	1	2	4	-0.45407429983	2	2	1	-0.45052415132	1
		2word16	3	2	-0.18899750304	1	3	3	-0.4346028884	0	3	3	-0.52489585375	0
		basic16	4	7	-0.45303574889	3	4	7	-0.579615909	3	4	4	-0.57099747919	0
		3class12	5	4	-0.28309796453	1	5	2	-0.33753384767	3	5	5	-0.58017546055	0
		2word12	6	6	-0.31745131738	0	6	5	-0.49108150848	1	6	7	-0.61490864585	1
		comp8	7	5	-0.2965234856	2	7	6	-0.54963875987	1	7	8	-0.65135140868	1
		basic12	8	8	-0.47954187505	0	8	8	-0.58639470743	0	8	6	-0.59158613934	2
	Extraneous	3class16	1	1	-0.04210526403	0	1	1	-0.1732211426	0	1	2	-0.25848766731	1
		basic20	2	4	-0.15048415667	2	2	3	-0.2410656647	1	2	1	-0.2478302857	1
		2word16	3	2	-0.05134151255	1	3	4	-0.2463640901	1	3	3	-0.29777195789	0
		basic16	4	6	-0.17558403806	2	4	7	-0.38191467167	3	4	4	-0.40219971884	0
		3class12	5	5	-0.15869415661	0	5	2	-0.22171184179	3	5	6	-0.43333756896	1
		2word12	6	7	-0.18670016936	1	6	6	-0.3512831245	0	6	5	-0.42869639987	1
		comp8	7	3	-0.15048415667	4	7	5	-0.29031771829	2	7	7	-0.4594561195	0
		basic12	8	8	-0.35504148566	0	8	8	-0.49858696195	0	8	8	-0.53008440019	0
	Convergent	3class16	1	7	-1.33181526992	6	1	2	-0.73706003039	1	1	5	-0.84807451306	4
		basic20	2	8	-1.65587234842	6	2	7	-0.86310442053	5	2	8	-0.90303209873	6
		2word16	3	6	-1.33177869336	3	3	5	-0.79623023624	2	3	7	-0.89905475536	4
		basic16	4	5	-1.02369206677	1	4	6	-0.85713632354	2	4	3	-0.79663046993	1
		3class12	5	2	-0.77450139244	3	5	1	-0.66271940447	4	5	2	-0.77997709357	3
		2word12	6	3	-0.82018732762	3	6	3	-0.74833314449	3	6	4	-0.83475601093	2
		comp8	7	4	-0.87004936668	3	7	8	-0.92869922291	1	7	6	-0.84854866977	1
		basic12	8	1	-0.77238736541	7	8	4	-0.77957401152	4	8	1	-0.73119269609	7