NISTIR 7970

Taxonomic Rules for Password Policies: Translating the Informal to the Formal Language

Kevin Killourhy
Yee-Yin Choong
Mary Theofanos

http://dx.doi.org/10.6028/NIST.IR.7970

National Institute of Standards and Technology
U.S. Department of Commerce

NISTIR 7970

Taxonomic Rules for Password Policies: Translating the Informal to the Formal Language

Kevin Killourhy
Yee-Yin Choong
Mary Theofanos
Information Access Division
Information Technology Laboratory

http://dx.doi.org/10.6028/NIST.IR.7970

December 2013

U.S. Department of Commerce
Penny Pritzker, Secretary

National Institute of Standards and Technology
Patrick D. Gallagher, Under Secretary of Commerce for Standards and Technology and Director

Table of Contents

1 INTRODUCTION

A password policy may seem formal in the sense that it is written in a legalistic language, giving the impression of a binding contract. However, such policies are informal in the logical sense that the policy statements are not written in a clear, unambiguous form. In password policy research at the National Institute of Standards and Technology, a formal language has been developed to explicitly capture what is expected of the user. This document presents that formal language grammar and the procedure that has been developed to translate the statements in the informal language of standard password policies to the formal language.[1]

This work is preliminary. As such, not all statements within every topic covered by any password policy are considered within scope. Instead, attention in this work is restricted to a subset of topics, acknowledging the remainder as presently beyond scope. The criteria used to differentiate between topics within and out of scope are given in **Sec. 2**. Further, the procedure for marking policy documents to identify within scope statements is described.

Once a statement is identified as within scope, the translation step prescribes that elements from the grammar are selected to construct one or more statements which capture the meaning of the informal statement regarding user behavior. A word-by-word translation is not performed. The grammar of the formal language is presented in **Sec. 3,** along with a description of the conventions used when translating ambiguous, informal statements into the formal language. The syntax of the formal language is in Extended Backus-Naur Form (EBNF)[2] notation, which is frequently used in programming language specification.

In **Sec. 4**, an example policy and its translation into the formal language is presented. A reader simply trying to understand the general concept of this translation process and not the details of the formal language branching, might want to review this section first. It should provide an overview of the process without the amount of grammar detail presented in **Sec. 2** and **Sec. 3**.

---

[1] The reference to any commercial products in this document is not intended to imply recommendation or endorsement by National Institute of Standards and Technology, nor is it intended to imply that the products are necessarily the best available for the purpose.

[2] ISO/IEC 14977 : 1996(E)

2 IDENTIFYING WITHIN SCOPE STATEMENTS

Upon initial perusal, many password policies make similar types of statements. Policies contain length and character set requirements, restrictions on how passwords are stored and whether they can be communicated, and warnings about consequences for a user who enters a password incorrectly too many times. While individual requirements, restrictions, and consequences may differ slightly, the impression is that all policies cover largely similar territory.

However, upon closer examination, differences emerge in the topics covered by different policies. Some policies regulate applications' and system administrators' behavior as well as users'. Some policies discourage passwords entirely, recommending either a security token or long passphrases with dramatically different composition requirements. Some policies lay out multiple sub-policies, each pertaining to passwords for different kinds of accounts (e.g., shared, normal user, privileged user, or system accounts).

2.1 WITHIN SCOPE TOPICS

Rather than attempt to capture the full diversity of the topics covered in password policies, a subset of topics that commonly appear among many different policies are identified as within scope. Particular attention is paid to the subset that covers those topics which most closely specify user behavior. These expectations on user behavior are expressed as explicit statements about what a user must, must not, should, and should not do. The topics cover (1) creating a password, (2) communicating a password, (3) changing a password, (4) storing a password, and (5) failing to authenticate.

Currently the focus is restricted to one account per policy: the general user account. When it is unclear which of multiple accounts corresponds to that of a general user, the least privileged, non-guest account is selected. All other account types are considered out of scope.

2.2 OUT-OF-SCOPE TOPICS

The following topics were judged to be presently beyond scope:

1. **Passphrases, PINs, and Alternatives:** Some policy writers use passphrase and password synonymously, while others define a passphrase to be longer than a password (e.g., 15 or more characters), created by assembling a sequence of words (not characters). This latter notion of a passphrase is certainly worth study, especially as several security experts have started to recommend them, but the focus of this research is restricted to passwords for now. Likewise, personal identification numbers

(PINs) for personal digital assistants (e.g., Blackberries) are beyond scope. More generally, alternatives to passwords are not yet included in the formal language. For example, policy statements that explicitly discourage password use in favor of one time passwords or public keys. Such statements are ignored.

2. **Non-standard Accounts and Access:** A single policy can regulate behavior for several different kinds of accounts. For instance, a policy might specify one length and lifetime for general users, a stricter length and lifetime for users with elevated privileges (e.g., those who handle personally identifying information), and require documented permission from management to maintain a group account with a shared password. All account types which do not correspond to the general user account are considered out of scope.

3. **Non-User Agents:** A password policy might specify the behavior for many parties, not just the general user. For instance, system administrators might be required to change passwords or disable accounts when users are terminated or transferred to another group. Applications may be required to obscure passwords when they are typed rather than displaying them on the screen. While formalized password policy statements could have many applications, they were formulated to help us express the expectations and responsibilities placed on users. Consequently, regulations of the behavior of these non-user agents are beyond scope.

4. **Infrequent Topics:** Some policy topics are considered infrequent, either in the sense that they are covered in very few policies, or they specify behavior which arises rarely. The consequences of forgotten passwords and expired accounts fall into this category. Many policies do not cover these topics explicitly so their study is postponed for future work. Likewise, initial password procedures are beyond scope (e.g., how new accounts are assigned passwords and when they must be changed). Statements about such procedures regulate behavior that is only relevant once per account, and they typically involve the activity of non-user agents, for example the rules pertaining to the default password for a new account.

5. **Platitudes and Generic Advice:** General tips and tricks for creating passwords are not translated into the formal language. Choosing a password that is easy to remember but hard to guess is easier said than done. Guidance about creating acronyms from phrases or converting letters to numbers may help users choose good passwords but they are suggestions not regulations. Only tips with explicit recommendations or prohibitions are translated (e.g., using mixed case letters or avoiding proper nouns).

The formal password policy language could be extended to express statements about these currently out-of-scope topics. To do so for the present work seemed premature. If the current work proves useful, such extensions might be warranted.

2.3 MARKING WITHIN SCOPE STATEMENTS

To facilitate the translation process, the following supporting process is used. The translator examines the document to find sentences, bullets, or other spans of text judged to be within scope regulating users' behavior when creating or managing passwords. Each such statement to be translated is identified and the translator inserts a parenthetical number in the document margin, starting at (1) and incrementing the number for each identified span of text. To make the identifiers stand out. In this document they are formatted as red text. These parenthetical indices are used to link translated statements back to statements in the source document.

3 TRANSLATING INTO THE FORMAL LANGUAGE

Having identified those policy statements that are within scope in **Sec. 2**, this section describes how these statements are translated into the formal password policy language.

As a formal language, there are absolute rules (called a grammar) regarding how policy statements must be phrased. In the same way that a misplaced word or punctuation mark might cause a syntax error in a programming language, an incorrectly phrased policy statement will violate the rules of the policy grammar. This grammar has been specified in a concise, unambiguous notation called Extended Backus-Naur Form (EBNF). The notation is commonly used to describe the syntax of programming languages. EBNF was adopted because tools exist to parse and automatically check the correctness of a document written with an EBNF-specified grammar.

Throughout this section, the formal rules of the EBNF grammar are presented alongside descriptions of what the rules mean. Readers unfamiliar with EBNF notation should still find the syntax of the rules fairly intuitive. First, the overall structure of policies and statements is described. Then, each kind of statement is discussed.

3.1 STRUCTURE OF A PASSWORD POLICY

Formally, a password policy is a collection of statements about behavior. This rule formally defines a policy to be a collection of one or more policy statements (`policy_stmt's`). The parenthetical `(s)` denotes one or more policy statements. It is denoted by the following rule:

```
1  policy: policy_stmt(s)
```

Each policy statement is decomposed into the following parts:

```
policy_stmt: rule_idx "Users" modal_verb pwd_stmt '.'
```

A policy statement is divided into a rule index (`rule_idx`), and a sentence with "`Users`" as the subject, a modal verb phrase (`modal_verb`), a statement about passwords (`pwd_stmt`), and a period (".") to terminate the sentence.

The rule index is defined by the following rule:

```
rule_idx: /\d+\w*/ ":"
```

The index of the rule is intended to connect the formal policy statement with a location in the informal policy document for later reference. It is encoded as a number followed by zero or more letters and then a colon. The expression `/\d+\w*/` is a UNIX-style regular expression denoting one or more digits followed by zero or more letters. For instance, "`1:`" and "`2a:`" are valid rule indices while "`a2:`" and "`1`" (with no colon) are not.

The numeric portion of the rule index is intended to correspond with the number marked (according to **Sec. 2.3**) next to the informal statement being translated. The alphabetic portion is intended to differentiate multiple formal statements that correspond to the same informal statement.

The substance of a policy statement is the sentence that follows the rule index. The subject of the sentence is "`Users`," and the subject is followed by one of four modal verb phrases according to the following rule:

```
modal_verb: "must not"
     | "should not"
     | "must"
     | "should"
```

The rule uses a vertical bar (`|`) to denote a choice among four options. Each option has its standard meaning. For instance, a sentence beginning "`Users must not`" is prohibiting behavior, and a sentence beginning "`Users should`" is recommending behavior.

Sometimes it can be tricky to determine whether an informally written policy is forbidding an activity (`must not`) or merely discouraging it (`should not`). When deciding, look for strong statements (e.g., "Never use a dictionary word") as evidence of a requirement or prohibition. However, if such statements are wrapped in weaker wording (e.g., "Consider the following tips for creating a strong password:"), view the statement as a recommendation. Note that sometimes a policy will present tips for creating a strong password and later clarify that these tips must be followed. Careful reading is often required.

After the modal verb comes the verb phrase that explicitly defines the behavior regulated by the statement:

```
pwd_stmt: change_stmt
     | commun_stmt
     | create_stmt
     | store_stmt
     | failauth_stmt
```

As with modal verb, the vertical bars denote a choice. In this password statement, the choice is between five different topics: (1) changing a password, (2) communicating a password, (3) creating a password, (4) storing a password, and (5) failing to authenticate. Each of these five kinds of statements have different structures and are described in separate sections.

3.2 RULES ON PASSWORD LIFETIMES AND EXPIRY

Statements that regulate when users must change their password have the following form:

```
change_stmt: "change passwords" time_crit
     | "change passwords" time_crit "if" event_name
```

The change statement (`change_stmt`) is either an absolute or a conditional statement. The absolute statement requires only a time criteria (`time_crit`) as defined by the following rule:

```
time_crit: "immediately"
     | "before" day_name
```

In the password policies that were examined as part of this research, passwords must either be changed immediately or before a certain number of days have passed. The number of days must be specified according to the `day_name` rule:

```
day_name: /1 day/
     | /\d+\s+days/
```

As an example of the use of the absolute change statement, if an informal policy stated, “Passwords must be changed every 90 days,”. It would be translated as follows:

```
1: Users must change passwords before 90 days.
```

The conditional statement is used to express the fact that, if a certain event (`event_name`) takes place, the password must be changed according to a specified time criteria. The kinds of events described in password policies are captured by the following rule:

```
event_name: "compromised"
```

```
2       | "directed by management"
3       | "found non-compliant"
4       | "sent unencrypted"
5       | "shared"
```

An informal policy statement might not correspond exactly to one of these events, but the match should be close to one. For instance, an informal policy might state, "Passwords that have been compromised or are suspected of being compromised must be changed right away." Such a statement would translate into the formal statement:

```
2: Users must change passwords immediately if compromised.
```

3.3 RULES ON PASSWORD COMMUNICATION AND TRANSMISSION

Statements concerning whether users talk about or transmit their passwords (e.g., over the Internet) are defined by the following rule:

```
1   commun_stmt: "communicate passwords" "to" person_name
2       | "communicate passwords" "by" media_name
3       | "communicate passwords" "except in an emergency"
```

In accordance with this rule, communications regulations take one of three forms. The first form governs to whom a user can communicate a password. Only two sets of people (`person_name`) appear in the password policies that were examined:

```
1   person_name: "a third party"
2       | "anyone"
```

For instance, an informal statement such as "Do not share your password with friends, family, secretaries, etc." would be translated to the statement:

```
3: Users must not communicate passwords to anyone.
```

The second form governs the transmission medium (`media_name`) by which a user can or cannot communicate passwords:

```
1   media_name: "any means"
2       | "any network without encryption"
3       | "any network"
4       | "email without encryption"
5       | "email"
6       | "mail without encryption"
7       | "mail accompanied by the user ID"
8       | "mail"
9       | "phone mail"
10      | "phone"
11      | "Internet or wide-area network without encryption"
```

```
12      | "Internet or wide-area network"
13      | "local-area network without encryption"
14      | "local-area network"
```

As in other cases, a translator may need to make a judgment about which of the choices is most appropriate in a given translation. For instance, the informal statement, "Passwords on the Internet must be encrypted in compliance with FIPS 140-2," would be translated as follows:

```
4: Users must not communicate passwords by Internet or wide-area network
without encryption.
```

The third form exists as a special case. Several policies make an exception to an overall ban on communicating passwords in the event of an emergency or other extraordinary circumstances. Such exceptions are translated into the formal language as:

```
5: Users must not communicate passwords except in an emergency.
```

3.4 RULES ON PASSWORD CREATION AND COMPOSITION

The majority of statements about passwords govern their length, the characters that must (and must not) appear, and other choices that are made when the user creates the password. Statements about password creation are defined by the following rule:

```
1  create_stmt: "create passwords" "with length" charlen_cmp
2       | "create passwords" /with a character in the first \d+ characters/
        charset_crit
3       | "create passwords" "with a character" charset_crit
4       | "create passwords" /with \d+ or more characters/ charset_crit
5       | "create passwords" "with all characters" charset_crit
6       | "create passwords" "with an internal character" charset_crit
7       | "create passwords" "with a first or last character" charset_crit
8       | "create passwords" "with a substring" stringset_crit
9       | "create passwords" stringset_crit
```

These nine different kinds of password creation requirement basically fall into one of three classes: length (line 1), character set (lines 2–7), and string set (lines 8–9). Each class is considered separately.

3.4.1 Rules on password length

Formal statements about password length include the phrase "`create passwords with length`" followed by a character length comparison (`charlen_cmp`) defined with:

```
1  charlen_cmp: "greater than or equal to" char_length
```

The only comparison is an inequality, requiring a character length (`char_length`) defined:

```
1  char_length: /\d+\s+characters/
```

Any number followed by spaces followed by the word "`characters`" is a valid length. Comparisons are expressed with inequalities of the single form "`greater than or equal to`" because all other forms of inequality and equality can be derived from it. This restriction prevents two different formal statements from expressing equivalent expectations about user behavior. For instance, the informal statement "Passwords must be between 8 and 20 characters" can only be translated as:

```
6a: Users must create passwords with length greater than or equal to 8
characters.

6b: Users must not create passwords with length greater than or equal to
21 characters.
```

Note that one informal statement may need to be translated as multiple formal statements. These different formal statements are denoted by assigning each one a different letter in the rule index (i.e., 6*a* and 6*b*).

3.4.2 Rules on password character sets

Statements about password character sets regulate which characters are allowed (or recommended) at which positions within a password. The six different character set choices (lines 2–7 in the `create_stmt` definition) lay out six different sets of positions that are regulated. For instance, consider line 2 which regulates `/a character in the first \d+ characters/`, and again, the pattern `\d+` means any number is valid. This choice would be used to capture a statement requiring that certain kinds of characters be in the first 7 positions in a password. Such statements appear in some policies.

All six choices of creation statement which regulate character sets include a character set criteria (`charset_crit`) defined as follows:

```
1  charset_crit: "in the set of" charset_name
```

The only criteria concerns membership in a character set (`charset_name`), defined as:

```
1  charset_name: basic_charset(s /,/)
```

All character sets that were examined were combinations of one or more basic types of character set (`basic_charset`), which are specified as a comma separated list. These basic character sets are:

```
1  basic_charset: "upper-case letters"
2       | "lower-case letters"
3       | "letters (unspec)"
4       | "numbers"
5       | /these special characters: .+ /
6       | "special characters (unspec)"
7       | "punctuation"
8       | "control or non-printable characters (unspec)"
9       | "whitespace"
10      | "all other characters (unspec)"
11      | "those not used in the previous password"
12      | "those used in the previous password"
```

The characters contained in each basic character sets are often evident. However, in some cases, where the actual set of characters is ambiguous the "`(unspec)`" tag is included in the name. The tag makes the underspecified nature of the character set explicit. Statements involving special characters are often underspecified, as in "special characters (e.g., , $, %)." Except for these examples, the characters in the set are not specified. In those cases where a full set of special characters is provided, the set can be captured using line 5 of the `basic_charset` rule. Note that the list of special characters must be separated on both sides by whitespace. For instance, the informal statement "Passwords must have one or more letters (upper or lower case), digits (which can't be in the first or last position), and the following special characters: `!@#$%^&*()`" would be translated as

```
7a: Users must create passwords with a character in the set of upper-case
letters, lower-case letters.

7b: Users must create passwords with an internal character in the set of
numbers.

7c: Users must create passwords with a character in the set of these
special characters: !@#$%^&*().
```

Note that, by convention, when a character set includes multiple basic character sets separated by a comma, they are ordered as listed above (e.g., "upper-case letters" comes before "lower-case letters").

Regarding translations involving special characters, the terms "special character," "symbol," and "punctuation" are considered to be equivalent, *except* when punctuation is explicitly differentiated from other kinds of non-alphanumeric characters. In such cases, and only in

such cases, is the punctuation character set used. For instance a requirement for "numbers, punctuation, or all other characters" would be translated as "`numbers, punctuation, all other characters (unspec)`."

3.4.3 Rules on password string sets

Some policies regulate not just characters in the password but whole sequences of characters (or strings). As in line 8 of the `create_stmt` rule, these statements can involve any substring of the whole password or, as in line 9, they can involve only the whole password. Both kinds of statements about sets of strings require a string set criteria (`stringset_crit`), defined as

```
1  stringset_crit: "in the set of" stringset_name
2       | "equal to" string_name
```

Two kinds of criteria were found in the password policy review, equality and set membership. The equality choice requires a string name (`string_name`):

```
1  string_name: "the user ID"
2       | "their name"
```

Only two strings have been regulated, the user ID and the user's name. For instance, the informal policy statement, "Do not include your user ID in your password" would be translated as

```
8: Users must not create passwords with a substring equal to the user ID.
```

More commonly, set membership regulations are seen. These statements involve a set of strings (`stringset_name`) defined by the following rule:

```
1  stringset_name: "addresses or other locations"
2       | "birthdays or other dates"
3       | "dictionary words" "followed by a number"
4       | "dictionary words" "in reverse"
5       | "dictionary words" "with numbers substituted for letters"
6       | "dictionary words" "preceded or followed by a number or special
        character (unspec)"
7       | "dictionary words"
8       | "incremental changes to existing passwords (unspec)"
9       | "otherwise forbidden content" "concatenated"
10      | "otherwise forbidden content" "in reverse"
11      | "otherwise forbidden content" "preceded or followed by a number"
12      | "passwords" "to an outside system"
13      | "passwords" "to any other system"
14      | "proper nouns"
```

```
15      | "personally identifying information"
16      | "strings with" /a character repeated \d+ or more times
        consecutively/
17      | "strings with" /a character repeated \d+ or more times/
18      | "strings with" /a run of \d+ or more consecutive characters in
        sequence/
19      | "strings with" /at least \d+ unique characters/
20      | "strings with" /characters from \d+ of these \d+ sets:/
        charset_name
21      | "strings with word or number patterns (unspec)"
22      | "their last" /\d+ passwords/
23      | "their last" /\d+ years of passwords/
24      | /those passwords used \d+ times in the last \d+ years/
25      | "vendor default passwords"
```

Many of the string sets are self-explanatory. Rather than splitting hairs, any prohibition against using names—family, pet, sports teams, or fantasy character—are grouped under the set of `proper nouns`. Likewise, different policies forbid words from different dictionaries, but the current grammar ignores such nuances and groups all such sets as `dictionary words`. For instance, the informal statement "Your password cannot be a word in English, Japanese, or Klingon, either forward or in reverse" would be translated as:

```
9a: Users must not create passwords in the set of dictionary words.

9b: Users must not create passwords in the set of dictionary words in
reverse.
```

These sets were found to satisfactorily capture the sets that appear in the password policies that were examined. In the future, as new policies are examined, this rule might need to be expanded to allow more choices.

3.5 RULES ON WRITING AND STORING PASSWORDS

Statements that regulate where and how users can store their passwords have the following form:

```
1   store_stmt: "store passwords" "in writing" loc_name
2       | "store passwords" "online" loc_name
```

Two kinds of storage are regulated: written and online. Both cases require a location name (`loc_name`) to explain the regulation. The location name is defined as follows:

```
1   loc_name: "anywhere"
2       | "in a secure location"
3       | "in an insecure location"
4       | "in automated scripts"
```

```
5       | "in clear text or weakly encrypted"
6       | "in clear text in an insecure location"
7       | "on outside systems"
```

Some policies forbid writing a password anywhere, others forbid writing them in clear text, and still others forbid writing them in clear text and storing that clear text password in an insecure location. Each of these different regulations is expressed as a different location (`loc_name`).

Once again, a translator must decide which of the choices in the formal grammar most closely correspond to the informal statement. For instance, consider the statements "You should never write your password in a place where others might find them," and "Passwords must be stored at a protection level no lower than that of the information protected by the password." With the current grammar, these statements would be translated as follows:

```
10a: Users must not store passwords in writing in an insecure location.

10b: Users must not store passwords online in an insecure location.
```

Prohibitions on using "remember password" features of web browsers or other applications fall under the "`in automated scripts`" choice. The umbrella location "`in clear text or weakly encrypted`" is meant to include restrictions on "clear text" with no mention of weak encryption.

3.6 RULES ON FAILED AUTHENTICATIONS AND LOCKOUT

Password lockout statements can be viewed from a user's perspective as an expectation (i.e., a limit on the number of incorrect authentication attempts) and a consequence for failing to meet that expectation (e.g., lockout for 15 minutes). Statements relating to authentication failure are formally coded using the following grammar rule:

```
1  failauth_stmt: "fail to authenticate" rep_name "to avoid" lockout_name
```

They require a repetition name (rep name) which is defined as:

```
1  rep_name: /\d+ times in a \d+ \w+ interval/
2       | /\d+ times/
```

The policies examined specify lockout either as a threshold number of incorrect logins (assumed consecutive) or a threshold number in a given time interval. The consequences of the lockout for the user (`lockout_name`) are defined as follows:

```
1  lockout_name: /administrative unlock or a \d+ \w+ lockout/
2       | "administrative unlock"
3       | "a lockout of unspecified duration"
```

```
4       | /a \d+ \w+ lockout/
```

The user is locked out of the account until either some time period (specified or not) has passed or a system administrator performs some action to unlock the account.

For instance, if a policy states, “Lockout occurs after 3 attempts; Users must present their ID at the help desk to have their account unlocked and their password reset,” it would translate it as

```
11: Users must not fail to authenticate 3 times to avoid administrative
unlock.
```

As with all the translation rules, some amount of careful reading and minor approximation may be necessary when rendering the ambiguities and vagaries of informal rules in an explicit and formal language.

4 EXAMPLE

To illustrate the process of translating an informal password policy into the formal policy language, an example is presented next. At each step, an explanation is given regarding the translation rationale.

4.1 IDENTIFYING WITHIN SCOPE STATEMENTS

The following policy was obtained from NASA’s Entry Descent Landing Repository (and modified slightly for the sake of example). It is not one of the policies in the corpus examined, but it has many of the same structures, making it illustrative.

```
1       Users shall adhere to the following password protections:
2       -Upon first login, the user account password shall be changed
3    (1) -Users shall not share account passwords
4    (2) -Passwords shall be between 8 and 31 characters in length, where
5       supported by the operating system
6    (3) -Passwords shall contain at least one character each from at
7       least three of the following sets of characters: uppercase letters,
8       lowercase letters, numbers, special characters, where supported by
        the
9       operating system.
10   (4) -Passwords shall not be a dictionary word.
11   (5) -Passwords shall not be either wholly or predominantly composed
12      of the following: The user’s ID, owner’s name, birth date, Social
13      Security Number, family member or pet names, names spelled backwards,
14      or other personal information about the user.
15   (6) -Passwords shall not be the name of a vendor, product,
16      contractor, project, division, section or group.
```

```
17 (7) -Passwords shall not be repetitive or a keyboard pattern.
18 (8) -Passwords shall not be the name of an automobile, sports team,
19     athlete, or other popular cultural symbols.
20 (9) -Passwords shall not be any of the precluded categories with
21     numbers appended or prepended.
22 (10)-A user account for system access shall have used a minimum of 24
23     passwords before a password can be reused.
24 (11)Note that User account passwords will expire after 90 days. Two
25     notices will be sent via email to the user alerting them to the
26     upcoming expiration. If the password is allowed to expire, you
27 (12)must contact the EDLR Curator to enable the account. Passwords
28     shall not be reused before 2 years have elapsed.
```

The first step in translating this policy is to identify which statements are within scope, as discussed in **Sec. 2**. Line 1 does not concern a specific behavior and so is beyond scope. Line 2 concerns initial passwords which are beyond scope per the *Infrequent Topics* item in **Sec. 2**. Line 3 is the first specific within scope statement, concerning password communication (though there is some ambiguity as will be discussed below). Since it is within scope, it is marked with a (1) in the left margin. All of the remaining bullets in lines 4–23, are also within scope, so they are marked with an incrementing sequence of numbers in the margin. The first sentence in line 24 is within scope (a change requirement), while the second sentence that begins on that line is not (regulating application behavior). The sentence beginning on line 26 concerns the consequence of an expired password and is beyond scope (again as in *Infrequent Topics*). The final sentence, beginning on line 27, is within scope (concerning password creation). In total, 12 statements were judged to be within scope and marked. In the example policy above, the within scope statements have been enumerated in red, in the left margin, just as a translator would do when reading the PDF version of the policy.

4.2 TRANSLATING INTO THE FORMAL LANGUAGE

Statement (1) states that users must not share passwords. Note an apparent ambiguity. Some translators might interpret this statement to mean that users should not tell others what their passwords are. Other translators might fairly interpret the statement to mean that users must not use the same password across multiple accounts. With no way to resolve this ambiguity with complete satisfaction, the former interpretation was chosen by the translator, and created the following translation:

```
1: Users must not communicate passwords to anyone.
```

Statement (2) regulates user behavior when creating passwords. Specifically, the statement sets a minimum and a maximum length. Note that the final clause concerns non-user agents (i.e., the operating system), so it is beyond scope. The resulting translation follows:

```
2a: Users must create passwords with length greater than or equal to 8
characters.

2b: Users must not create passwords with length greater than or equal to
32 characters.
```

Statement (3) is directly translated into another password creation requirement:

```
3: Users must create passwords in the set of strings with characters from
3 of these 4 sets: upper-case letters, lower-case letters, numbers,
special characters (unspec).
```

Note that the character sets are listed in the same order they appear in the `basic_charset` definition. Since the set of special characters is not defined, the *`(unspec)`* tag is used.

Statement (4) is translated as:

```
4: Users must not create passwords in the set of dictionary words.
```

Statement (5) restricts the password content in a variety of ways, at a level of specificity not included in this formal language. For instance, "pet names" is not among the string sets. However, all the restrictions are part of the more general set of *personally identifying information*, and so the spirit of the statement can be translated as:

```
5: Users must not create passwords with a substring in the set of
personally identifying information.
```

Note that the specifying phrase "not [...] predominantly composed of" is interpreted to mean "`must not create passwords with a substring in the set of`."

Statement (6) also provides a level of specificity that is not captured in these rules. Since all of the forbidden things are proper nouns, the statement is translated as:

```
6: Users must not create passwords in the set of proper nouns.
```

Statement (7) is directly translated as:

```
7: Users must not create passwords in the set of strings with word or
number patterns (unspec).
```

Because there are many different kinds of patterns and repetitions and the statement does not unambiguously explain what patterns are disallowed, the *`(unspec)`* tag is used.

Statement (8) provides additional proper names which are forbidden, and so it is translated as a duplicate of statement 6:

```
8: Users must not create passwords in the set of proper nouns.
```

By convention, duplicate rules may or may not be translated. If the duplicate statement was marked as within scope, the duplicate translation is created because every marked statement must have a corresponding translation. However, during markup, if the translator notices that a statement is repeated more than once, the translator has the option of not marking every instance; a choice that would be influenced by analysis objectives.

Statement (9) can be directly translated as:

```
9: Users must not create passwords in the set of otherwise forbidden
content preceded or followed by a number.
```

Statement (10) translates as:

```
10: Users must not create passwords in the set of their last 24 passwords.
```

Statement (11) translates as:

```
11: Users must change passwords before 90 days.
```

Statement (12) translates as:

```
12: Users must not create passwords in the set of their last 2 years of
passwords.
```

To understand how these translations conform to the rules of the formal grammar, it is suggested that new translators take each of the translated rules and develop a parse tree. Identify the `modal_verb` and the `pwd_phrase`; determine which kind of statement it is (e.g., `create_stmt` or `change_stmt`); subdivide those statements into their components (e.g., `event_names` and `stringset_names`). By checking that each formal statement complies with the grammar, one might develop an intuition about how to compose such statements.

5 SUMMARY

In this document, a formal grammar for password polices specifying the expectations for general users is described, as well as how to translate informal password policies into that formal language. The intent was to describe this translation process in such a way that a new translator might use the description to develop this skill. The first step is to read the policy, and to identify and mark statements which are within scope. The second step is to consider each of the marked statements and compose one or more statements written in the formal policy language that express the same expectation on user behavior. This second step is guided by the syntax of the language (i.e., which formal statements are permissible and

which are not), and by a discussion of the considerations when statements are selected from among the set of permissible statements.