# Sharing Susceptible Passwords as Cyber Threat Intelligence Feed

Iman Vakilinia*, Sui Cheung†, Shamik Sengupta‡
Department of Computer Science and Engineering, University of Nevada, Reno
Reno, NV, USA
Email: *ivakilinia@unr.edu, †scheung@unr.edu, ‡ssengupta@unr.edu

***Abstract*—Password-strength checkers provide feedback to users about their password choice. Several parameters are investigated by password-strength checkers such as length, character set, user information, and entropy to score the chosen password. Moreover, such checkers use dictionaries to detect susceptible passwords such as keyboard sequences (e.g. `qwert`), simple and usual words (e.g. `password`). As the password patterns are language specific, having an English dictionary of patterns is not helpful to detect other language patterns. Besides that, the users' passwords choice might be inspired by the new patterns emerging in their culture. For instance, new movies, books, and games. Hence, the dictionary needs to be updated to cover the new patterns. However, generating such dictionaries which cover new and diverse patterns is not simple and needs excessive efforts to extract new patterns from different cultures.**

**To update the list of susceptible passwords and extract new password patterns dynamically, we propose a method for sharing attacked passwords which are collected from the honeypot through brute-force attack attempts. To achieve this goal, first, we analyze the passwords collected in brute-force attack attempted in our honeypot. Then, we model the password sharing as cyber threat intelligence feed in the Structured Threat Information Expression (STIX) format. We also provide a tool which allows users to query if a string or its leet transformation is existing in the susceptible password dataset.**

***Index Terms*—Password, Cybersecurity Information Sharing, Threat Intelligence**

## I. Introduction

Password-based authentication is the most common method for authentication services, despite the recent increased development in public-key infrastructure and multi-factor techniques. According to Verizon's 2017 Data Breach Investigations Report, 81% of hacking-related breaches leveraged either stolen and/or weak passwords [1]. To overcome this issue, servers apply password policies to prohibit users from choosing simple passwords. For instance, password policies usually enforce a minimum length for the passwords. Moreover, one of the well-known password requirement is to check if the password's character set contains Lower and Uppercase letters, Digits and Symbols known as *LUDS*. However, there are many research studies that showed the insufficiency of *LUDS* [2]–[4] password requirements. For example, studies have shown that users tend to use particular characters more than others. Additionally, *LUDS* does not check passwords that contain the most common patterns. Last but not least, *LUDS* rejects a large set of strong passwords which do not contain specific characters (e.g. `DkRPSjx`).

In addition to policies for password requirements, there are password-strength checkers that compute the password strength considering several factors and it assists users to find a better password. Servers may prevent a password to be selected if its score is less than a threshold. Password-strength checkers calculate a score based on several parameters such as length, character set, user information, and entropy. Moreover, the password-strength checker uses a dictionary to detect susceptible passwords such as simple and usual words (e.g. `password`) and easy keyboard sequences (e.g. `qwert`). Since password patterns are language specific, using an English dictionary is not as helpful as to detect other language patterns [5]. Furthermore, a user's password choice might be inspired by the new patterns emerging in their cultures such as new movies, books, animations, and games. Hence, the word dictionary needs to be updated to cover the new patterns. However, generating such word dictionaries is not an easy task and it requires excessive efforts to extract new patterns from different cultures.

On the other hand, attackers also apply various dictionaries to perform brute-force attacks on password-based authentication systems. Such dictionaries convey a wide range of words with different levels of complexity. Furthermore, these word lists are not limited to one language or culture and it is more dynamic compared to currently available password dictionaries.

To proactively defend against attackers, cyber threat intelligence has been proposed. Gartner defines cyber threat intelligence as: "evidence-based knowledge, including context, mechanisms, indicators, implications and actionable advice, about an existing or emerging menace or hazard to assets that can be used to inform decisions regarding the subjects response to that menace or hazard." [6]. The indicator of Compromise (*IoC*) is the most common cyber threat feeds. It includes blacklist IP addresses, malware signatures, malicious URLs and domain names. However, such information is not sufficient to protect password-based authentication systems.

As the available password-strength checkers do not update the list of susceptible passwords and they are insufficient to classify the passwords, we propose a method for sharing updated susceptible passwords which have been used by the attackers. To achieve this goal, first, we set up a honeypot to collect attackers' password lists and then we present those passwords as cyber threat feed for the password-strength checkers.

TABLE I: Cybersecurity information sharing specifications

| Specification | Description |
|---|---|
| STIX | Structured Threat Information Expression is a language and serialization format for exchanging cyber threat intelligence |
| TAXII | Trusted Automated eXchange of Indicator Information is a free and open transport mechanism that standardizes the automated exchange of cyber threat information |
| CybOX | Cyber Observable eXpression is an expression format which has been integrated into STIX 2.0 |
| VERIS | The Vocabulary for Event Recording and Incident Sharing is a set of metrics designed to provide a common language for describing security incidents in a structured and repeatable manner |
| MAEC | Malware Attribute Enumeration and Characterization is a standardized language for sharing structured information about malware based upon attributes such as behaviors, artifacts, and attack patterns |
| IODEF | Incident Object Description Exchange Format is a data format which is used to describe computer security information for the purpose of exchange between Computer Security Incident Response Teams |
| OpenIOC | OpenIOC has information about the indicators of compromise which includes threats details, attack methodologies, and information about the vulnerable platform. |

We model the password sharing in the Structured Threat Information Expression (*STIX*) format [7] which is a language and serialization format for exchanging cyber threat intelligence (*CTI*). Furthermore, we provide a tool for users to check if a string and its leet transformations exist in the attackers' password dataset. The contribution of this work is as follows.

- We analyze the attackers' dataset of passwords by comparing them with existing password datasets. To this end, we have deployed a low-interaction honeypot to collect attackers' passwords which have been applied in the brute-force attacks.
- We propose a method for sharing the susceptible passwords as cyber threat intelligence feeds to improve the password-strength checkers' performance.

The rest of the paper is organized as follows. The next section reviews major works in the password-strength checkers and cybersecurity information sharing. Section IV, describes our investigation on collected passwords from brute-force attacks performed on our honeypot. Details of our proposed sharing method are described in Section V. Finally, we conclude the paper in Section VI.

## II. Related Work

Estimating the password-strength have been studied extensively for many years both experimentally and theoretically. Applying dictionary words to impose password rules have been studied in [8]–[10]. Authors in [3] analyzed the attacker's guess-ability of passwords by evaluating different attacks over the dataset of passwords. Li et al. [5] investigate the distinction between password inclination of Chinese and English users. The effectiveness of applying password-strength checkers has been studied in [11], [12]. Furnell [13] assessed the password guidelines and policies of several well-known web service providers and evaluated the enforcement of such requirements.

Several research studies have been done for designing the robust password meters [2], [14]–[16]. `zxcvbn` [2] is the open-source password-strength estimator tool which is used by Dropbox. In its core, `zxcvbn` investigates the generality of a password based on several sources which are common passwords, leaked passwords, common names, and common words from Wikipedia. Authors in [17] evaluate and characterize the password meters of several popular websites such as Dropbox, eBay, Google, Microsoft, and Paypal. *Telepathwords* [18] have been proposed to examine passwords against guessing attacks such that it makes a real-time prediction of the next character as the user types a password.

Furthermore, several research studies have analyzed the passwords in brute-force attacks and discuss the defensive techniques [19]–[21]. Moreover, they argue that based on the collected password dataset, what is known as a strong password is not sufficient to protect systems authentication since the attackers' passwords dictionary conveys the so-called strong passwords.

On the other hand, recently, a large number of research studies focused on the cyber threat intelligence and cybersecurity information sharing [22]–[24]. To provide a common platform for sharing such information, various protocols and specifications for cybersecurity information sharing have been developed which are listed in Table I [25]–[28]. Several repositories provide threat intelligence feeds such as *Virus Total* [29], *CYMON* [30], and *HAIL A TAXII* [31]. The threat intelligence feeds are *IoC*s such as blacklist IP addresses, malware signatures, malicious URLs and domain names. However, such information is not sufficient to protect password-based authentication systems.

Inspired by the previous works, in this paper, we analyze the passwords which are currently being used in the brute-force attacks. Then, we propose a method for sharing information about such passwords to improve the password-strength checkers for the detection of susceptible passwords.

## III. System Architecture

In this section, we elaborate the honeypot installation for collecting the attackers' passwords lists.

To collect the passwords used by the attackers on the Internet, we have installed two low-interaction honeypots with the subsequent IP addresses. Although brute-force attacks are running over many services such as *Telnet*, *FTP*, *SSH*, *Email*, and *web*, in our data collection we are focusing on *SSH* brute-force attempts because it has the highest rate and the collected dataset conveys the other services brute-force passwords.

We have changed the code of *OpenSSH server 7.6* to log the *username*, *password*, and *source IP address* in the login attempts. This change has been done in `auth_password` function located in `auth-passwd.c` file by adding the following line of code:

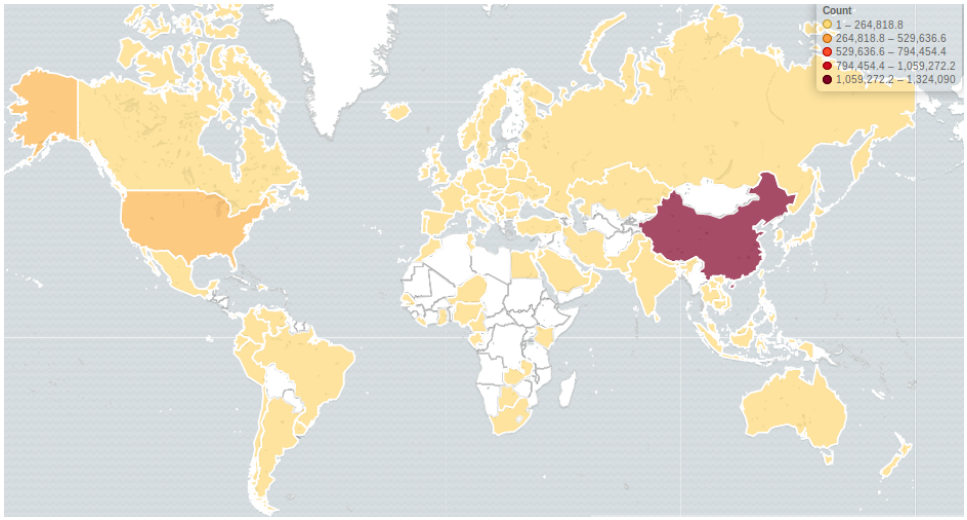


Fig. 1: Geomap of the source IPs initiating the SSH brute-force attacks using *Kibana*

```
logit("IP: %s Port: %d Username: %s Password:
%s", ssh_remote_ipaddr(active_state),
ssh_remote_port(active_state), authctxt->user,
password);
```

We also use *ELK*[1] stack 6.1 to parse, transfer, store and query the collected information. Once *SSH* login attempt happens, *rsyslog*[2] sends the login information to *ELK*. To this end, we have set the *rsyslog* config file as follows:

```
:msg, regex, "IP:.*Port:.*Username:.*Password:
.*" @Logstash_IP:Logstash_port
```

*Logstash* first receives, parses and tags data, and then *Logstash* transfers them to *Elasticsearch*. *Elasticsearch* stores and indexes data, and finally *Kibana* is used to query and visualize the collected information. Figure 2 shows the system architecture. We use *QEMU* and *KVM* for virtualization and *pfSense*[3] as our gateway and firewall. We have put honeypots in a separate network by deploying of the virtual switches.

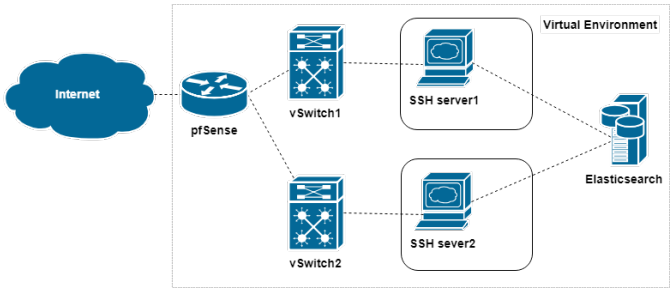


Fig. 2: System architecture

## IV. Passwords Collection and Analysis

In this section, we analyze the collected passwords by comparing them to the available password datasets. Then we check the scores that several passwords are receiving from available password-strength checkers.

[1]*Elasticsearch Logstash Kibana, https://www.elastic.co/products*
[2]http://www.rsyslog.com/
[3]https://www.pfsense.org/

The data has been collected from the first of November-2017, until the end of January-2018. Figure 1 shows the geolocation map of the source IP addresses that initiated the *SSH* brute-force attacks. Table II shows the overall information about the collected logs.

TABLE II: Overall information about SSH brute-force attacks

| | **Testbed-1** | **Testbed-2** |
|---|---|---|
| Total login Attempts | 1,018,101 | 838,109 |
| Number of Source IP addresses | 3,063 | 2,144 |
| Number of Usernames | 1,062 | 805 |
| Number of Passwords | 69,427 | 68,202 |

From the total login attempts, the most frequent username that has been attempted is *"root"* and the most frequent password is *"123"* with $126,134$ times. Excluding *"123"*, Figure 3 displays the top $40$ passwords that have been tried by attackers in our dataset. The highest number of login attempts happened from one IP address, was $69$ times.

In total, we have gathered $69,644$ distinct passwords from both datasets. The longest password is "*0xLsHymi8BFU353Qr0wctoIGlSwnLfSInQo24S0qWlkZS5jb20vZG93b00 wctoIG0*" which has $65$ characters.

By analyzing our datasets, we can see that attackers have different strategies. For example, one class of attackers have a small list of passwords (usually less than ten passwords), and they have tried those passwords at different times. Another set of attackers have a large password dataset and choose several passwords each time they perform brute-force. On the other hand, it can be seen that there are groups of attackers which share the same password datasets. Botnets are usually using the same password dataset. Although it is not sufficient to conclude one specific botnet based on just username and password login attempts, we suspect that a large number of requests to our honeypot have been initiated by the Mirai botnet, since this botnet has $62$ pair of username and password which are existing in our dataset and have been applied by the attackers repetitively [32].

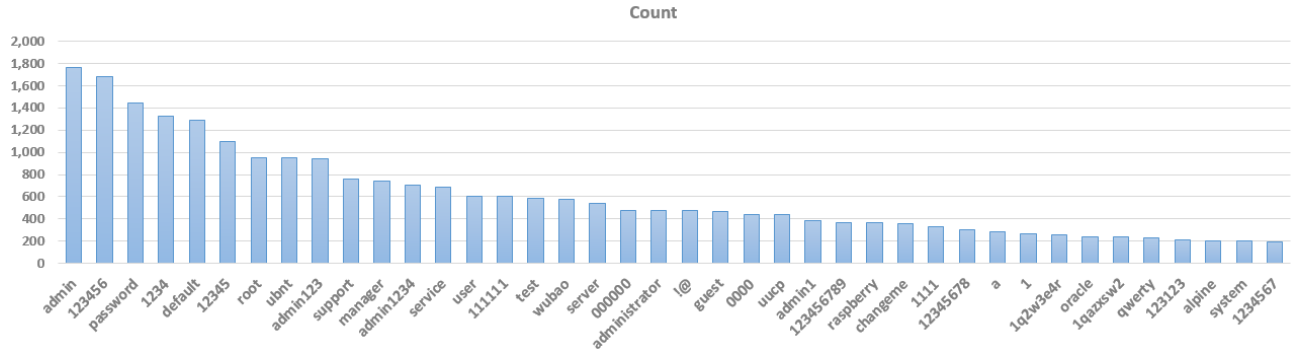


Fig. 3: Top 40 passwords excluding "123"

We have used three public available password datasets[4] of *Top_passwords*, *phpBB*, and *RockYou*, to compare with our dataset. *Top_password* is the list of $100,000$ most common passwords, *phpBB* and *RockYou* are two leaked password databases from *phpBB.com* and *RockYou.com* websites, respectively. These lists include $184,389$, and $14,344,382$ passwords, respectively. Comparing our datasets with these three datasets, Figure 4 shows how many passwords in our attack dataset are not existed in any of these datasets. Figure 5 shows the number of passwords with various length. As it can be seen there are a large number of passwords exist with larger than six characters. This indicates that the length policy is good, but it is not enough to resist against brute-force attacks.

From the collected passwords, $1,399$ passwords following the *LUDS* policy. We have investigated the score of multiple passwords by using the multi-checker tool provided by MADIBA security research group [17]. Table III depicts the results of password strength scores calculated by Apple, Dropbox, eBay, Google, Drupal, Intel, PayPal, QQ, Twitter, Yahoo!, and Yandex. As it can be seen, most of the password-strength checkers are giving relatively high scores to the selected passwords. However, these passwords have been tried several times by the attackers. Among the service providers, Dropbox is the only one which has presented the algorithm of its password-strength checker [2]. Our experiment supports Carnavalet and Mannan research [17] that the Dropbox password-strength checker generates a more realistic classification mainly because of checking how common a password is according to several datasets. However, as these datasets are not completely covering all of the words and are not dynamically filled, this password-strength checker is also incapable to realistically classifying many passwords.

For example, consider "$SeRvEr2003@$" as password, this password receives "$Weak2/5$" from Dropbox, while as you can see in the table "$FuWuQi2003@$" receives "$Great!5/5$" from Dropbox. Other password-checkers also classify this password as strong. Whereas *"Fuwuqi"* is the Chinese translation of *"Server"* and it has appeared in attack dataset multiple times. This case is true for other passwords as well, for example as you can see the second password contains *"Mima"* which is the Chinese translation of *"Password"*. The third password contains word *"Spadie"* which is the name of a Japanese poker league. Moreover, we have noticed a large number of website addresses have been tried by attackers. By checking these passwords, the majority of password-strength checkers classify them as strong since they are not equipped with a dictionary of such website addresses. Thus, we can see that the password-strength checkers are not efficient in classifying a large set of passwords considering the attack datasets.

[4]Downloaded from https://github.com/danielmiessler/SecLists

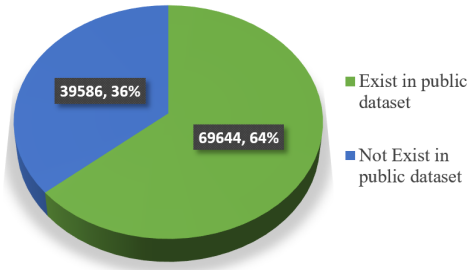


Fig. 4: Number of unique passwords in our dataset

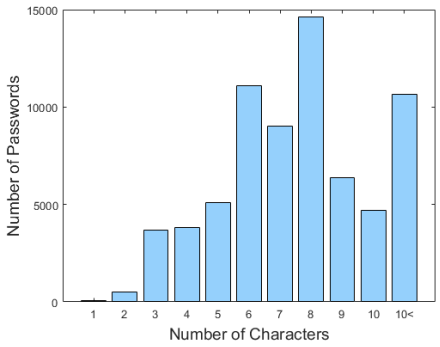


Fig. 5: Number of passwords of different length

## V. Passwords as Threat Intelligence Feeds

In this section, we present two methods for sharing information about the attackers' dictionary. In the first method, we use *STIX* format to transfer new password trends as cyber threat intelligence feed. In the second method, we provide a tool which allows users to query a string from the attackers' password dataset to check if there are passwords consisted of the string or its *leet* transformations in the attacker dictionary.

TABLE III: The classification of different passwords by several password-strength checkers

| | FuWuQi2003@ | 12345Mima!@ | SpAd!e-8 | AKG450@ikki.me | w!ns@ckP@$$12 | www.hao123.com |
|---|---|---|---|---|---|---|
| Apple | Strong 3/3 | Strong 3/3 | Moderate 2/3 | Strong 3/3 | Strong 3/3 | Moderate 2/3 |
| Dropbox | Great! 5/5 | Weak 2/5 | Good 4/5 | Great! 5/5 | Good 4/5 | Great! 5/5 |
| eBay | Strong 4/4 | Strong 5/5 | Strong 5/5 | Strong 5/5 | Strong 5/5 | Medium 4/5 |
| Google | Strong 5/5 | Strong 5/5 | Strong 5/5 | Strong 5/5 | Strong 5/5 | Strong 5/5 |
| Drupal | Strong 4/4 | Strong 4/4 | Strong 4/4 | Strong 4/4 | Strong 4/4 | Strong 4/4 |
| Intel | Oh No! 1/2 | Oh No! 1/2 | Oh No! 1/2 | Congratulations! 2/2 | Congratulations! 2/2 | Congratulations! 2/2 |
| PayPal | Strong 4/4 | Weak 2/4 | Strong 4/4 | Strong 4/4 | Strong 4/4 | Fair3/4 |
| QQ | Strong 4/4 | Strong 4/4 | Strong 4/4 | Strong 4/4 | Strong 4/4 | Strong 4/4 |
| Twitter | Perfect 6/6 | Perfect 6/6 | Perfect 6/6 | Perfect 6/6 | Perfect 6/6 | Okay5/6 |
| Yahoo! | Very Strong 4/4 | Very Strong 4/4 | Very Strong 4/4 | Very Strong 4/4 | Very Strong 4/4 | Strong 3/4 |
| Yandex | OK 4/4 | OK 4/4 | OK 4/4 | OK 4/4 | OK 4/4 | OK 4/4 |

TABLE IV: Leet Transformations available in the dataset

| String | Leet Transformations available in the dataset |
|---|---|
| Root | ROOT, Root, rOOt, ro0t, root, ro0t, r0ot, r00t, ro0t, 2007 |
| Admin | ADMIN, ADmin, AdmIn, Adm1n, Admin, adm!n, adm1n, admin, 4dm1n, 4dmin, @dm1n, @dmin |
| Pass | PASS, Pass, P@SS, P@ss, pass, p@ss, p@$$, 6425 |
| Server | SERVER, Server, server, s3rv3r |
| User | User, user, us3r, u5er, u53r, uzer |
| Mima | MIma, Mima, mima |
| Fuwuqi | FUWUQI, FuWuQi, Fuwuqi, fUWUqI, fuwuqi |


```
{
  "type": "observed-data",
  "id": "observed-data--71ed3e65-
      8844-4f3a-9a49-d321d4a2cd79",
  "created_by_ref": "identity--4a127483-
      292d-4dee-9680-b3897beed6e8",
  "created": "2017-12-14T18:26:14.134Z",
  "modified": "2018-01-01T18:26:14.213Z",
  "first_observed": "2017-12-14T18:26:14.134Z",
  "last_observed": "2018-01-01T18:26:14.213Z",
  "number_observed": 3,
  "objects": {
    "0": {
      "type": "password",
      "value": "FuWuQi2003!"
    }
  }
}
```



Fig. 6: STIX Observed Data to represent attacker password

### A. Using STIX to share weak passwords

*STIX* components are Objects and Relationships. *STIX* Objects represent information with a set of attributes. Relationships chain multiple objects to simplify the representation of complex *CTI*.

We use STIX Observed Data object to represent password information. Observed Data object provides information about the system and network elements using cyber observable specification [33]. For example, the observation of an IP address, an email, a file, or a registry key can be represented by STIX Observed Data object.

Irrespective of the data type an Observed data object has some common properties: *type*, *id*, *created_by_ref*, *created*, *modified*, *first_observed*, *last_observed*, and *number_observed*. The *type* field must be "observed-data" as it describe the object we are representing. *id* and *created_by_ref* utilizes Universally Unique Identifiers (UUID) to identify the particular event and the organization respectively. *Created*, *modified*, *first_observed*, and *last_observed* are timestamps in Coordinated Universal Time (UTC) zone and are self-explanatory. *number_observed* field identifies the number of occurrence of the event within the time period from *first_observed* to *last_observed*. This allows the observed data object to aggregate multiple sightings of the same event and consecutively reduce data clutter. Figure 6 depicts an example of our proposed *STIX* presentation for sharing a susceptible password. Here STIX Observed Data object indicates that the password *"FuWuQi2003!"* has been observed three times over the time window of *2017-12-14T18:26:14.134Z* till *2018-01-01T18:26:14.213Z*.

Note that since many passwords in the attack dataset are completely random, in order to increase the data quality, it is better to filter passwords before publishing them. For filtering, we consider a threshold value $\tau$ as a requirement for publishing. In this case, a password is going to be reported if it is repeated by different IP addresses for $\tau$ times in a time window (e.g. from Jan/01/2018 till March/01/2018). Also, this filtering can happen on the client side which is the receiver of the shared information. In this case, a receiver can put a password in its blacklist if it receives the password from different sources and with a predefined threshold.

### B. Searching among the attack dataset

*Leet* transformation replaces characters with other characters which are visually similar. For instance, the letter "*i*" is analogous to "1", "!", "\", "/", and ";" characters. Users usually apply the *leet* transformation to make their password complex and meet the password policy requirements such as *LUDS*. We provide a tool which checks if a string or its *leet* transformations, is in the list of attackers' datasets or not. This helps users to have a better understanding of guess-ability of their passwords to avoid susceptible passwords. For this purpose, our tool receives a string as input and queries all

of its *leet* transformations from the attacker password dataset collected by the honeypot.

The password-checker receives a string as input and finds every password which has the string or its *leet* transformations as a substring in a password dataset. First, the checker computes all of the *leet* transformations. Then, it searches for the passwords that contained a substring of each transformation. As an example, the Table IV depicts the available passwords transformations for the input strings of "root", "admin", "pass", "server", "user", "mima", and "fuwuqi". in our collected password dataset.

The scripts and the collected password dataset are in a Github repository and available at `https://github.com/imanvk/SSH-BruteForce-Honeypot` for research purpose.

## VI. Conclusion and Future Works

Password-strength checkers help users to choose safer passwords by scoring the chosen password. However, such checkers do not consider the passwords that have been used by the attackers for brute-force. Making a honeypot to collect attack passwords dataset, we modeled sharing susceptible password as cyber threat feeds using STIX. We have also provided a tool that allows users to query a string from a password dataset to check if the string and its *leet* transformations exist as a substring of vulnerable passwords. For the future works, We will measure the rate of appearance of a new password in the dataset. Having such information, we would then investigate what kind of algorithm attackers use to extract new passwords. Also, we will investigate if it is possible to classify botnets based on their password datasets.

## VII. Acknowledgment

This research is supported by the National Science Foundation (NSF), USA, Award #1739032. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

## References

[1] "Verizon 2017 data breach report," http://www.verizonenterprise.com/resources/reports/rp_DBIR_2017_Report_execsummary_en_xg.pdf.

[2] D. L. Wheeler, "zxcvbn: Low-budget password strength estimation." in *USENIX Security Symposium*, 2016, pp. 157–173.

[3] M. Dell'Amico, P. Michiardi, and Y. Roudier, "Password strength: An empirical analysis," in *INFOCOM, 2010 Proceedings IEEE*. IEEE, 2010, pp. 1–9.

[4] R. Shay, S. Komanduri, P. G. Kelley, P. G. Leon, M. L. Mazurek, L. Bauer, N. Christin, and L. F. Cranor, "Encountering stronger password requirements: user attitudes and behaviors," in *Proceedings of the Sixth Symposium on Usable Privacy and Security*. ACM, 2010, p. 2.

[5] Z. Li, W. Han, and W. Xu, "A large-scale empirical analysis of chinese web passwords." in *USENIX Security Symposium*, 2014, pp. 559–574.

[6] "Threat intelligence," https://www.gartner.com/imagesrv/media-products/pdf/webroot/issue1_webroot.pdf.

[7] "Structured threat information expression (stix)," https://oasis-open.github.io/cti-documentation/.

[8] D. V. Klein, "Foiling the cracker: A survey of, and improvements to, password security," in *Proceedings of the 2nd USENIX Security Workshop*, 1990, pp. 5–14.

[9] J. Nagle, "An obvious password detector," *USENET news*, vol. 16, p. 60, 1988.

[10] M. Bishop, "Anatomy of a proactive password changer," *group*, vol. 501, p. 20, 1992.

[11] B. Ur, P. G. Kelley, S. Komanduri, J. Lee, M. Maass, M. L. Mazurek, T. Passaro, R. Shay, T. Vidas, L. Bauer *et al.*, "How does your password measure up? the effect of strength meters on password creation." in *USENIX Security Symposium*, 2012, pp. 65–80.

[12] S. Egelman, A. Sotirakopoulos, I. Muslukhov, K. Beznosov, and C. Herley, "Does my password go up to eleven?: the impact of password meters on password selection," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2013, pp. 2379–2388.

[13] S. Furnell, "Assessing password guidance and enforcement on leading websites," *Computer Fraud & Security*, vol. 2011, no. 12, pp. 10–18, 2011.

[14] C. Castelluccia, M. Dürmuth, and D. Perito, "Adaptive password-strength meters from markov models." in *NDSS*, 2012.

[15] B. Rodrigues, J. Paiva, V. Gomes, C. Morris, and W. Calixto, "Passfault: an open source tool for measuring password complexity and strength," *Orlando, Florida, Mar*, 2017.

[16] M. Dell'Amico and M. Filippone, "Monte carlo strength evaluation: Fast and reliable password checking," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2015, pp. 158–169.

[17] X. D. C. De Carnavalet, M. Mannan *et al.*, "From very weak to very strong: Analyzing password-strength meters." in *NDSS*, vol. 14, 2014, pp. 23–26.

[18] S. Komanduri, R. Shay, L. F. Cranor, C. Herley, and S. E. Schechter, "Telepathwords: Preventing weak passwords by reading users' minds." in *USENIX Security Symposium*, 2014, pp. 591–606.

[19] J. Owens and J. Matthews, "A study of passwords and methods used in brute-force ssh attacks," in *USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET)*, 2008.

[20] A. Abdou, D. Barrera, and P. C. Van Oorschot, "What lies beneath? analyzing automated ssh bruteforce attacks," in *International Conference on Passwords*. Springer, 2015, pp. 72–91.

[21] E. Kheirkhah, S. M. P. Amin, H. A. Sistani, and H. Acharya, "An experimental study of ssh attacks by using honeypot decoys," *Indian Journal of Science and Technology*, vol. 6, no. 12, pp. 5567–5578, 2013.

[22] I. Vakilinia, D. K. Tosh, and S. Sengupta, "Privacy-preserving cybersecurity information exchange mechanism," in *Performance Evaluation of Computer and Telecommunication Systems (SPECTS), 2017 International Symposium on*. IEEE, 2017.

[23] I. Vakilinia and S. Sengupta, "A coalitional game theory approach for cybersecurity information sharing," in *Military Communications Conference, MILCOM 2017-2017 IEEE*. IEEE, 2017.

[24] I. Vakilinia, D. K. Tosh, and S. Sengupta, "3-way game model for privacy-preserving cybersecurity information exchange framework," in *Military Communications Conference, MILCOM 2017-2017 IEEE*. IEEE, 2017.

[25] "Standards and tools for exchange and processing of actionable information," https://www.enisa.europa.eu/publications/standards-and-tools-for-exchange-and-processing-of-actionable-information/.

[26] J. Steinberger, A. Sperotto, M. Golling, and H. Baier, "How to exchange security events? overview and evaluation of formats and protocols," in *Integrated Network Management (IM), 2015 IFIP/IEEE International Symposium on*. IEEE, 2015, pp. 261–269.

[27] P. Kampanakis, "Security automation and threat information-sharing options," *Security & Privacy, IEEE*, vol. 12, no. 5, pp. 42–51, 2014.

[28] C. Wagner, A. Dulaunoy, G. Wagener, and A. Iklody, "Misp: The design and implementation of a collaborative threat intelligence sharing platform," in *Proceedings of the 2016 ACM on Workshop on Information Sharing and Collaborative Security*. ACM, 2016, pp. 49–56.

[29] "Virus total," https://www.virustotal.com/.

[30] "Cymon," https://www.cymon.io/.

[31] "Hail a taxii," http://hailataxii.com/.

[32] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis *et al.*, "Understanding the mirai botnet," in *USENIX Security Symposium*, 2017.

[33] "Stix version 2.0. part 4: Cyber observable objects. edited by trey darley and ivan kirillov." 19 July 2017. OASIS Committee Specification 01. http://docs.oasisopen.org/cti/stix/v2.0/cs01/part4-cyber-observable-objects/stix-v2.0-cs01-part4-cyber-observableobjects.html. Latest version: http://docs.oasis-open.org/cti/stix/v2.0/stix-v2.0-part4-cyberobservable-objects.html.