"Will Any Password Do?" Exploring Rate-Limiting on the Web

Maximilian Golla*, Theodor Schnitzler*, and Markus Dürmuth
Horst Görtz Institute
Ruhr-University Bochum
{maximilian.golla, theodor.schnitzler, markus.duermuth}@rub.de

ABSTRACT

We empirically analyzed whether and how real-world websites take appropriate measures to prevent unauthorized accesses to their users' accounts. We tried to get access to our own accounts on 12 different services, pretending to have forgotten our password and entering alternatives before taking further measures. Our findings indicate that providers' measures to counter trawling online guessing attempts widely differ. We faced CAPTCHAs, temporal blocking, and lockouts from our accounts. We observed that large services combine many mechanisms. In the trade-off between security and usability smaller sites lock down accounts and involve their users. We even observed a service that didn't rate-limit at all, which burdens users with strong passwords.

1. INTRODUCTION

Common advice given to users for protecting their online accounts is to use strong passwords, not to re-use a password for more than one account, not to write it down, and ideally to change it frequently [30]. These recommendations require substantial cognitive effort and are virtually impossible to follow for end-users [25] as shown by the escalating password reuse problem [29]. This observation has started a process to adapt recommendations to more realistic advice [27]. For example, to group accounts and reuse passwords within the same category [17], or no longer recommending to frequently change a password [28].

As a countermeasure against weak passwords, online services implement rate-limiting to prevent online guessing attacks. Rate-limiting comes in a variety of flavors. Typically, the number of failed login attempts per account is counted, and if it rises beyond a threshold, services slow down verification of subsequent login attempts. Moreover, it can require the user to solve CAPTCHAs or provide additional authentication factors, notify the user of suspicious behavior, or just lock the account, either silently or with notification.

*The first two authors contributed equally to the paper.

Who Are You?! Adventures in Authentication (WAY) 2018.
August 12, 2018, Baltimore, MD, USA.

In this paper, we take a closer look at real-world implementations of rate-limiting, and the effects on strength requirements for online passwords. In more detail: First, we explore if and how a sample of 12 larger websites from a set of diverse categories implements rate-limiting in practice. We model a realistic attack scenario to observe the website's behavior. We find a wide range of implementation choices, e. g., when rate-limiting takes effect, how the attack is slowed down, and if users are notified or required to take additional steps. Second, our findings provide evidence that password strength is not the (only) driving factor for account security, and that rate-limiting may provide an alternative, and potentially even better solution against password guessing than stronger passwords.

The results presented in this paper indicate that the weak memorability of strong passwords can be traded for tight rate-limiting on the server side. This potentially has a significant impact on the usability of password-based authentication on online services. However, strict rate-limiting likely has usability problems as well, which are, to the best of our knowledge, not well understood. Fingerprinting techniques and risk-based authentication might help in reducing the burden for the user. Furthermore, it is important that these findings only hold for online services, where rate-limiting and throttling can reliably be implemented. It does not at all hold for scenarios where offline guessing can be applied, e. g., for password managers, hard-disk encryption, and often for device unlock.

Outline In Section 2 we give an introduction to the related work in the area of rate-limiting. In Section 3 we explain our methodology and describe our study procedure. In Section 4 we present the results and discuss the implications of the findings, and conclude in Section 5.

2. RELATED WORK

In the following, we introduce the relevant terminology and give an overview of helpful related work.

2.1 Password and Account Security

It is challenging for a user to estimate the strength of a password [35]. Common methods like *password strength meters* [15], *password composition policies* [32], and *blacklists* [21] are used to nudge or force a user to create passwords that provide a reasonable level of security. Another option to reinforce the security of accounts is *two-factor authentication* (2FA). While usability studies [14] found that users believe that 2FA is easy to use and enabling it made

their account more secure, adoption of multi-factor authentication is still low, and users often complain (depending on the contexts of use) that 2FA is annoying.

Nowadays, major web services start to utilize suspicious activity detection [18, 26] and try to limit the success rate of so-called *credential stuffing attacks*, i. e., attacks that exploit the password reuse behavior of users [29], by proactively searching for credential leaks of third-party services [34].

2.2 Rate-Limiting and Throttling

The work that comes closest to our study is from Bonneau and Preibusch [9]. They reported on the poor rate-limiting deployed on the Web, even on high profile sites such as Amazon and eBay, in 2010. An overview of rate-limiting techniques (CAPTCHAs, blocking, and account locking) is given by Florêncio et al. [16]. As it is unlikely to find a single technology that perfectly solves all issues, a combination of rate-limiting mechanisms is required [7].

CAPTCHAs (for better readability, we write the acronym in lowercase) can be quite effective. However, they are susceptible to automated attacks [5, 33]. Nowadays, there are even crowdworker-driven automated solving services [19] that could be leveraged, depending on the attacker's budget. It remains a *cat-and-mouse game* between developers and attackers; furthermore captchas are often criticized for their usability on desktops [12] and smartphones [31]. There exist solutions that require answering fewer captchas for legitimate users [4].

Another option to throttle the number of illegitimate attempts is to temporarily *block access* (e. g., IP, cookie, or session-blocking). However, too restrictive policies, i. e., IP range-blocking, can degrade user experience and cause additional service desk calls [1]. For example, in preliminary tests, we found that many airlines block network traffic originating from the Tor network and request affected users to call the help desk. Moreover, botnets that are often used to execute guessing attacks are relatively wide-spread across the IP address space which escalates blocking and makes it less effective [24].

Even more severe is *account locking*. Here, the number of allowed guesses is not tied to the originating IP but the requested account. Exceeding a threshold causes the account to be locked, and requires either time, additional verification (e. g., via email or SMS), or even contact with the customer service to unlock the account again. While relatively secure, the biggest issue are lockout policies [10] that result in a denial of service vulnerability and a usability burden.

Relatively new, but already deployed by large services, are techniques known from risk-based authentication [18, 26] like browser fingerprinting [2].

3. METHODOLOGY

In this section, we explain our attacker model and the methodology of our study. We explain how we selected the websites we analyzed, the passwords we guessed and how we captured the rate-limiting employed by the service providers.

3.1 Attacker Model

In this work, we are concerned with *online guessing attacks*. In such an attack, the adversary is using a login form on a website or a similar mechanism to test different password candidates. Thus, the number of guesses an attacker can test is limited by the remote server. Typically, one considers between 100 and 1 000 allowed guesses within 30 days [20, 37] reasonable. If the correct password is tested, one will gain access otherwise one will be denied access.

In our study, we simulate a *trawling* attacker [8], i. e., the adversary is interested in the takeover of *any* account. Trawling attackers are well known in the context of, e. g., answering security questions [8]. Such attackers guess passwords in decreasing order of likelihood, i. e., most frequent passwords first, based on population-wide statistics [6]. More severe and starting to become more popular are *targeted* attacks, where an adversary is focused on a *specific* victim [37]. In such a scenario, personal information [13] and even leaked passwords [23] from breached services that (allegedly) belong to the victim are exploited to make more targeted guesses.

3.2 Selecting Websites

We wanted to model a realistic attack that also targets services that already make use of modern risk-based authentication [18, 26] techniques. To avoid an atypical influence of new accounts without history and value on such systems, we opted to use existing accounts. To prevent any harm from persons not involved, we chose to guess passwords for our own accounts. We extracted the list of 249 domains from one of our password managers and grouped them into 15 disjoint sets, where we loosely followed the categorization by Pearman et al. [29]. Next, we limited the set to services with a global Alexa rank [3] below 5 000 and selected two services from each of the six largest groups. We strived to create a diverse sample, covering both major services but also smaller or medium-sized websites. The services we tested were *Amazon*, *Dropbox*, *Facebook*, *Google*, *Grammarly*, *IKEA*, *Netflix*, *Plex*, *Trainline*, *Twitter*, *Uber*, and *Yahoo*.

3.3 Selecting Passwords

The baseline for the passwords we tested was the *Pwned Passwords v2* list of 500 million breached passwords, recently published by Troy Hunt [22]. For each service, we created a custom list of the top 25 passwords accepted by the service in two steps: First, we removed all passwords that did not comply with the service's composition policy from our baseline. Second, we manually verified the validity of the remaining passwords in ascending order, i. e., we tested if each password was accepted by the service to create a new account, eventually reaching an individual list for every particular service. The second step was inevitable due to several services being non-transparent in presenting which passwords they accept, e. g., *Google* requires "8 or more characters," but `12345678` is not allowed to use. None of the custom lists contained the actual password used for our account at that particular service.

3.4 Measuring Rate-Limiting

For each account, we performed 25 login attempts and observed the provider's behavior. Choosing a number too high would comprise an attack, which is to a certain extent unethical – even if we are trying to attack our accounts – and might also be illegal. We chose a number greater than 10, which is recommended from a usability perspective [10], but well below 100, which is specified as an upper bound by NIST [20] to throttle online guessing attacks. We further

think that 25 attempts are sufficient to gain a first impression of how providers behave, without unnecessarily wasting their resources. All attempts were performed by manually entering the credentials into the login forms on the services' websites using the *Tor Browser*. We have chosen to use the Tor network as it provides a convenient option to obtain new IP addresses (required to circumvent session/IP blocking). Furthermore, we consider this to be a more realistic attack scenario (cf. Section 2). We captured the countermeasures providers put in place during our study and logged how they influenced our ongoing procedures. After the 25 adversarial login attempts, we immediately tried to log in into the tested account by entering the correct credentials and observed the provider's behavior. We tried to log in using the same Tor session as for the latest adversarial attempt if the authentication failed we also attempted to log in using a device and browser known by the service.

4. RESULTS

In this section, we report and discuss our findings.

4.1 Overview

In our study, we have observed that the rate-limiting applied by service providers to counter adversarial login attempts widely differ. We provide a summary of our findings in Table 1.

We aimed to perform 25 adversarial login attempts. However, two services locked down our accounts, IKEA after 7 attempts, and Grammarly after 13 attempts. For the 10 remaining services, we were able to conduct the full set of 25 guesses, which took between 3 and 19 minutes due to diverging realizations of the throttling employed by the site operators. After the 25^{th} attempt, we were able to log in into six accounts by entering the correct credentials – from the same Tor session without switching our IP address. We have denoted this with a check mark in the *Login* column in Table 1. For one service we tested – the British rail platform Trainline – we observed no rate-limiting throughout all 25 consecutive login attempts. We were able to log in into the account immediately afterward. It might, of course, be possible that our number of login attempts was still below Trainline's rate-limiting threshold, but we think that this service does not adequately protect its accounts. For all other services, we saw varying realizations with different levels of intensity. In the following, we describe how the services we tested used captchas, account locking, and blocking to protect user accounts. We further report if and how we were notified of adversarial activity and which further measures were put in place after logging in with the correct credentials.

4.2 CAPTCHAs

Five services leveraged captchas to challenge the adversary. Dropbox, Uber, and Yahoo required us to solve a captcha even *before* we could perform the first login attempt, which we attribute to the use of the Tor network. After solving the initial captcha, the further procedures could not have been more diverging. While there were no further captchas to be solved for Yahoo, Dropbox challenged us with consecutive captchas after every single login attempt. In several cases, we had to solve three to five captchas (Google's *reCAPTCHA*), which became (subjectively) more difficult over time. Uber presented two captchas after the 11^{th} and 22^{nd} attempt. Captchas were further employed by Amazon and Google. The first occurrence for Google was after the 11^{th} attempt, followed by the 14^{th} and then every other attempt.

Manually solving captchas was quite time-consuming (for Dropbox it took 19 minutes to perform 25 attempts and to subsequently successfully sign in), but it remains unclear how effective this measure is against a sophisticated adversary. Captchas can be quite easy to solve for machines [5, 33, 11], there are even crowdworker-driven automated solving services [19]. Especially for Dropbox, captchas were the only rate-limiting we could observe, and we were able to log in afterward from within the adversarial session. We do not think that Dropbox takes security lightly, but if there are no further mechanisms deployed (after 25 guesses) their account security would be at risk.

4.3 Account Locking

We observed account lockouts for four services. While IKEA and Grammarly explicitly locked our accounts, lockouts were non-obvious and not reported to us in the cases of Netflix and Facebook. For Facebook, we were able to perform login attempts without barriers, but the final attempt with our correct password was simply rejected like every previous attempt with a wrong password. The website's response to the correct password did not differ from its reaction to any wrong password. Even when logging in from another known device, Facebook denied access stating that there were too many incorrect login attempts. A successful login was possible approx. 10 minutes later using a known/trusted browser configuration. In the case of Netflix, logging in with the correct password was also refused – both from the Tor session and from a known device. However, the login was possible approx. one hour later, without Netflix providing us any further notice.

Locking the account is a quite effective mechanism in terms of account security. From a usability perspective, locking is annoying for the user, and it opens the door for Denial of Service attacks. This specifically applies to Grammarly and IKEA, which burden users to take additional steps such as using the fallback authentication mechanism. Facebook and Netflix comprise a much better solution in only setting a temporal lockout. If users did not try to log in within this time span, they would not be bothered by the incident, while their accounts remain continuously protected from unauthorized accesses. Moreover, both do well in presenting the same reject message after the correct or an incorrect password has been entered. For that reason (and if there are not other side-channels such as response timing), an attacker would not notice if having guessed the password successfully.

4.4 Blocking

Several services rate-limited by blocking our IP address after a certain number of failed logins, i. e., we had to switch our IP generating a new Tor identity to continue trying to log in. For example, Amazon did not allow us to enter credentials after any four consecutive failed logins from the same session. Twitter blocked our IP address after 16 failed login attempts, mentioning a 60 minutes ban for further attempts. However, from a different IP address, we could immediately continue to enter passwords. Netflix was quite creative in that it pretended to have "technical difficulties" making it

Table 1: Results Overview

Alexa	Service	Category	# Guesses	Time	Login	CAPTCHA	Lockout	Blocking	2nd Step	Notification
1	Google	Communication	25	10 Min.	✗	11,14,16,18,20,22,24	–	≤ 25	–	–
3	Facebook	Social Network	25	4 Min.	✗	–	≤ 25	–	–	–
7	Yahoo	Communication	25	5 Min.	✗	0	–	≤ 25	Email Code	Suspicious
12	Twitter	Social Network	25	4 Min.	✓	–	–	16	Phone No.	Sign-in, Suspicious
30	Netflix	Entertainment	25	7 Min.	✗	–	≤ 25	7,14,21,22,...,25	–	–
84	Amazon	Shopping	25	15 Min.	✓	15,19,23	–	4,8,12,16,20,24	Email Code	–
89	Dropbox	Software	25	19 Min.	✓	0,...,24	–	–	–	Sign-in
285	IKEA	Shopping	7	2 Min.	✗	–	7	–	–	Account Locked
664	Grammarly	Software	13	6 Min.	✗	–	13	9,11,13	–	–
992	Plex	Entertainment	25	7 Min.	✓	–	–	7,14,21	–	–
1220	Uber	Travel	25	9 Min.	✓	0,11,22	–	11,22	SMS Code	–
4333	Trainline	Travel	25	3 Min.	✓	–	–	–	–	–

impossible to further trying to log in. Their rate-limiting occurred after the 7^{th}, 14^{th}, 21^{st}, and every consecutive attempt, requiring us to switch the IP address. Their blocking was combined with the temporal account lockout, as "technical difficulties" also occurred when we entered correct credentials from a known device. However, it was not apparent after which particular attempt the lockout was put in place.

Similarly, it was also not apparent at which point Yahoo's blocking took effect. We could freely enter 25 passwords for Yahoo, each one and also the correct password was simply rejected with the same message. When we switched to a new IP, logging in with the correct password was possible after entering a code Yahoo sent to us via an alternative email address. This indicates that adversarial behavior had been detected, and our first IP had been blocked without making this apparent. Similar to both, lockouts and blocking, we consider rate-limiting that silently takes effect the best option. In such a scenario, any further guesses are simply useless, but the attacker has no evidence for this and keeps trying. Even if the correct password is entered by chance, the login is denied, and the account remains protected.

4.5 Fingerprinting

The Tor browser warns its users when websites try to extract HTML5 canvas image data, which may be used to uniquely identify a computer. We observed such warnings on the login websites of Amazon, Facebook, Uber, and Yahoo. Additionally, we found fingerprinting data in HTTP requests of Google, Netflix, and Twitter. While we are unable to tell, whether the fingerprints are used for authentication decisions and rate-limiting or only for tracking and marketing, they are collected. For example the login request of Yahoo includes, besides IP, User Agent, and referrer, 22 client-side features that are generated by the *Fingerprintjs2* library [36]. The fingerprint includes language, window size, time zone, canvas-, local storage-, and WebGL support, hashes of installed plugins and fonts, as well as features like *has_lied*-language, resolution, OS, and browser.

4.6 Second Factors and Notifications

When finally logging in with the correct credentials, a few services required a second step, before we could successfully log in. Amazon and Yahoo emailed a code we had to enter, Uber sent a code via SMS. In the case of Twitter, we had to enter the phone number associated with the account. Dropbox informed us via email about a new sign-in after successfully logging in, Yahoo reported suspicious behavior, and Twitter did both. It is unknown whether such notifications contribute to security or rather raise uncertainty among users, for example in the case of false alerts (e. g., during traveling). If providers have successfully detected and blocked adversarial activity, bothering users with this could be considered questionable.

4.7 Limitations

Although the study was planned and conducted carefully, it comprises a few limitations related to how we guessed passwords. Due to limiting our study to only 25 guesses, we cannot provide an extensive review of rate-limiting in practice, but only provide a first impression of whether and how such mechanisms are put in place. Additionally, we simulated a rather weak attacker, in that we performed untargeted guessing by trying the top 25 passwords compliant with the services' policies. We have performed our guesses using the Tor network, however, it is not known how realistic it is for an attacker to use Tor. However, realistic adversaries would presumably act more sophisticated, e. g., exploiting password reuse, or spoofing the account owner's location by using an IP from a specific area or browser configuration. Furthermore, we only focused on the primary authentication interface. However, there are also reports of attacks exploiting a missing rate-limiting for the fallback authentication mechanism and API endpoints.

5. CONCLUSION

In this work, we explored practical implementations of rate-limiting in online services. Our results indicate that providers are aware of their responsibility to contribute to account security, instead of just forcing their users to set and remember strong passwords. There are differences in how providers apply rate-limiting. In total we have identified three levels of protection. First, large services combine mechanisms such as fingerprinting, captchas, blocking, multiple authentication steps, and notifications. Smaller websites lock down accounts and require manual account recovery involving their users and help desks. It is arguable whether involving users is an acceptable solution in the trade-off between security and usability (after an attack has been detected). However, leaving account protection solely to users by burdening them to use strong passwords is not recommendable.

6. REFERENCES

[1] A. Abdou, D. Barrera, and P. C. van Oorschot. What Lies Beneath? Analyzing Automated SSH Bruteforce Attacks. In *Conference on Passwords*, pages 72–91. Springer, 2015.

[2] F. Alaca and P. C. van Oorschot. Device Fingerprinting for Augmenting Web Authentication: Classification and Analysis of Methods. In *Annual*

Conference on Computer Security Applications, pages 289–301. ACM, 2016.

[3] Alexa Internet. Alexa: Traffic Statistics Global Ranking, May 2018. `https://www.alexa.com/siteinfo` , as of June 25, 2018.

[4] M. Alsaleh, M. Mannan, and P. C. van Oorschot. Revisiting Defenses against Large-Scale Online Password Guessing Attacks. *IEEE Transactions on Dependable and Secure Computing*, 9(1):128–141, 2012.

[5] K. Bock, D. Patel, G. Hughey, and D. Levin. unCaptcha: A Low-Resource Defeat of reCaptcha's Audio Challenge. In *Workshop on Offensive Technologies*. USENIX, 2017.

[6] J. Bonneau, C. Herley, P. C. van Oorschot, and F. Stajano. The Quest to Replace Passwords: A Framework for Comparative Evaluation of Web Authentication Schemes. In *Symposium on Security and Privacy*, pages 553–567. IEEE, 2012.

[7] J. Bonneau, C. Herley, P. C. van Oorschot, and F. Stajano. Passwords and the Evolution of Imperfect Authentication. *Communications of the ACM*, 58(7):78–87, 2015.

[8] J. Bonneau, M. Just, and G. Matthews. What's in a Name? Evaluating Statistical Attacks on Personal Knowledge Questions. In *Financial Cryptography and Data Security*, pages 98–113. Springer, 2010.

[9] J. Bonneau and S. Preibusch. The Password Thicket: Technical and Market Failures in Human Authentication on the Web. In *Workshop on the Economics of Information Security*, 2010.

[10] S. Brostoff and M. A. Sasse. "Ten Strikes and You're Out": Increasing the Number of Login Attempts can Improve Password Usability. In *Workshop on Human-Computer Interaction and Security Systems*. ACM, 2003.

[11] E. Bursztein, J. Aigrain, A. Moscicki, and J. C. Mitchell. The End is Nigh: Generic Solving of Text-based CAPTCHAs. In *Workshop on Offensive Technologies*. USENIX, 2014.

[12] E. Bursztein, A. Moscicki, C. Fabry, S. Bethard, et al. Easy Does It: More Usable CAPTCHAs. In *Conference on Human Factors in Computing Systems*, pages 2637–2646. ACM, 2014.

[13] C. Castelluccia, A. Chaabane, M. Dürmuth, and D. Perito. When Privacy Meets Security: Leveraging Personal Information for Password Cracking. *CoRR*, abs/1304.6584:1–16, 2013.

[14] J. Colnago, S. Devlin, M. Oates, C. Swoopes, et al. "It's Not Actually That Horrible": Exploring Adoption of Two-Factor Authentication at a University. In *Conference on Human Factors in Computing Systems*, pages 456:1–456:11. ACM, 2018.

[15] X. de Carné de Carnavalet and M. Mannan. From Very Weak to Very Strong: Analyzing Password-Strength Meters. In *Symposium on Network and Distributed System Security*. ISOC, 2014.

[16] D. Florêncio, C. Herley, and P. C. van Oorschot. An Administrator's Guide to Internet Password Research. In *Large Installation System Administration Conference*, pages 44–61. USENIX, 2014.

[17] D. Florêncio, C. Herley, and P. C. van Oorschot. Password Portfolios and the Finite-Effort User: Sustainably Managing Large Numbers of Accounts. In *Security Symposium*, pages 575–590. USENIX, 2014.

[18] D. M. Freeman, S. Jain, M. Dürmuth, B. Biggio, et al. Who Are You? A Statistical Approach to Measuring User Authenticity. In *Symposium on Network and Distributed System Security*. ISOC, 2016.

[19] M. Golla, D. V. Bailey, and M. Dürmuth. "I want my money back!" Limiting Online Password-Guessing Financially. In *Who Are You?! Adventures in Authentication Workshop*. USENIX, 2017.

[20] P. A. Grassi, J. L. Fenton, and W. E. Burr. Digital Identity Guidelines – Authentication and Lifecycle Management: NIST SP 800-63B, 2017.

[21] H. Habib, J. Colnago, W. Melicher, B. Ur, et al. Password Creation in the Presence of Blacklists. In *Workshop on Usable Security*. ISOC, 2017.

[22] T. Hunt. I've Just Launched "Pwned Passwords" V2 With Half a Billion Passwords for Download. `https://www.troyhunt.com/ive-just-launched-pwned-passwords-version-2/`, as of June 25, 2018.

[23] D. Logan. British Airways Among Latest Breaches. *Elsevier Network Security*, 2015(4):2–20, 2015.

[24] B. McCarty. Botnets: Big and Bigger. *IEEE Security & Privacy*, 1(4):87–90, 2003.

[25] R. McMillan. The Man Who Wrote Those Password Rules Has a New Tip: N3v$r M1-d! `https://www.wsj.com/articles/the-man-who-wrote-those-password-rules-has-a-new-tip-n3v-r-m1-d-1502124118`, as of June 25, 2018.

[26] G. Milka. Anatomy of Account Takeover. In *Enigma*. USENIX, 2018.

[27] National Cyber Security Centre. Password Guidance: Simplifying Your Approach. `https://www.ncsc.gov.uk/guidance/password-guidance-simplifying-your-approach`, as of June 25, 2018.

[28] National Cyber Security Centre. The Problems with Forcing Regular Password Expiry, Dec. 2016. `https://www.ncsc.gov.uk/articles/problems-forcing-regular-password-expiry`, as of June 25, 2018.

[29] S. Pearman, J. Thomas, P. E. Naeini, H. Habib, et al. Let's Go in for a Closer Look: Observing Passwords in Their Natural Habitat. In *Conference on Computer and Communications Security*, pages 295–310. ACM, 2017.

[30] R. W. Reeder, I. Ion, and S. Consolvo. 152 Simple Steps to Stay Safe Online: Security Advice for Non-tech-savvy Users. *IEEE Security & Privacy*, 15(5):55–64, 2017.

[31] G. Reynaga, S. Chiasson, and P. C. van Oorschot. Exploring the Usability of CAPTCHAS on Smartphones: Comparisons and Recommendations. In *Workshop on Usable Security*. ISOC, 2015.

[32] R. Shay, S. Komanduri, A. L. Durity, P. S. Huh, et al. Designing Password Policies for Strength and Usability. *ACM Transactions on Information and System Security*, 18(4):13:1–13:34, 2016.

[33] S. Sivakorn, I. Polakis, and A. D. Keromytis. I am Robot: (Deep) Learning to Break Semantic Image CAPTCHAs. In *European Symposium on Security and Privacy*, pages 388–403. IEEE, 2016.

[34] K. Thomas, F. Li, A. Zand, J. Barrett, et al. Data Breaches, Phishing, or Malware? Understanding the Risks of Stolen Credentials. In *Conference on Computer and Communications Security*, pages 1421–1434. ACM, 2017.

[35] B. Ur, J. Bees, S. M. Segreti, L. Bauer, et al. Do Users' Perceptions of Password Security Match Reality? In *Conference on Human Factors in Computing Systems*, pages 3748–3760. ACM, 2016.

[36] V. Vasilyev. Fingerprintjs2, May 2015. `https://valve.github.io/fingerprintjs2/`, as of June 25, 2018.

[37] D. Wang, Z. Zhang, P. Wang, J. Yan, et al. Targeted Online Password Guessing: An Underestimated Threat. In *Conference on Computer and Communications Security*, pages 1242–1254. ACM, 2016.

Implement a rate-limited queue with qrate. I wrote a Node.js module to help with the latter option. The qrate library lets you create queues and specify: concurrency â€” the number of jobs that will be worked on in parallel. rateLimit â€” the number of jobs per second that are allowed to be consumed from the queue. Hereâ€™s how it works. First, bring in the silverlining library to access a Cloudant database and the qrate package tooÂ If the qrate API looks familiar to you, then that is because it is based on the excellent async library, which provides a range of tools for writing asynchronous code for JavaScript. The qrate library is essentially the same as async.queue with an extra, optional rateLimit parameter. Without it, it behaves as a normal async.queue. Websites that rely on password-based login as their sole method of authenticating users can be highly vulnerable if they do not implement sufficient brute-force protection. Brute-forcing usernames. Usernames are especially easy to guess if they conform to a recognizable pattern, such as an email address.Â As the limit is based on the rate of HTTP requests sent from the user's IP address, it is sometimes also possible to bypass this defense if you can work out how to guess multiple passwords with a single request. LAB Broken brute-force protection, multiple credentials per request. HTTP basic authentication. Password reuse across websites remains a dire problem despite widespread advice for users to avoid it. Numerous studies over the past fteen years indicate that a large majority of users set the same or similar passwords across different websites (e.g., [8], [59], [65], [14], [39], [54], [70]).Â Alone, these methods do little to detect an attackerâ€™s use of a leaked, known-good password for one website at another website where the victim user is known to have an account.Â Limiting password-based access: Takada [68] proposed to interfere with the misuse of accounts with shared passwords by adding an â€œavailability controlâ€ to password authentication. In this design, a user disables the ability to log into her website account at a third-party service and then re-enables it when needed. I think I will be able to access it again shortly but regardless does anyone have any ideas as to what could have caused this? EDIT: It fixes itself (usually only a couple hours but times vary). What triggered it for me was spamming reactions on a discord message.Â " In the case that a rate limit is exceeded, the API will return a HTTP 429 response code with a JSON body. "" IP addresses that make too many invalid HTTP requests are automatically and temporarily restricted from accessing the Discord API. Currently, this limit is 10,000 per 10 minutes.Â got banned for the same reason, but it was a bot that made me do it. it's minigame was to click on the reaction as much as you could to get a certain amount of currency, ended up getting ip banned. ._. permalink. embed. 2do2go / rate-limit-mongo. Watch 3. Star 15. Fork 4. A MongoDB store for express-rate-limit middleware. 15 stars. 4 forks.Â Use Git or checkout with SVN using the web URL. Work fast with our official CLI. Learn more.Â var RateLimit = require('express-rate-limit'); var MongoStore = require('rate-limit-mongo'); var limiter = new RateLimit({ store: new MongoStore({ // see Configuration }), max: 100, windowMs: 15 * 60 * 1000 })Â password: string -- password for authentication in mongodb. authSource: string -- db name against which authenticate use. If not set db name from uri will be taken. collection: object -- mongodb collection instance. Required if uri hasn't been set.