A Measurement Study of Authentication Rate-Limiting Mechanisms of Modern Websites

Bo Lu*
The Ohio State University
lu.1294@osu.edu

Xiaokuan Zhang*
The Ohio State University
zhang.5840@osu.edu

Ziman Ling
The Ohio State University
ling.135@osu.edu

Yinqian Zhang
The Ohio State University
yinqian@cse.ohio-state.edu

Zhiqiang Lin
The Ohio State University
zlin@cse.ohio-state.edu

ABSTRACT

Text passwords remain a primary means for user authentication on modern computer systems. However, recent studies have shown the promises of guessing user passwords efficiently with auxiliary information of the targeted accounts, such as the users' personal information, previously used passwords, or those used in other systems. Authentication rate-limiting mechanisms, such as account lockout and login throttling, are common methods to defeat online password cracking attacks. But to date, no published studies have investigated how authentication rate-limiting is implemented by popular websites. In this paper, we present a measurement study of such countermeasures against online password cracking. Towards this end, we propose a black-box approach to modeling and validating the websites' implementation of the rate-limiting mechanisms. We applied the tool to examine all 182 websites that we were able to analyze in the Alexa Top 500 websites in the United States. The results are rather surprising: 131 websites (72%) allow frequent, unsuccessful login attempts without account lockout or login throttling (though some of these websites force the adversary to lower the login frequency or constantly change his IP addresses to circumvent the rate-limiting enforcement). The remaining 51 websites are not absolutely secure either: 28 websites may block a legitimate user with correct passwords when the account is locked out, effectively enabling authentication denial-of-service attacks.

CCS CONCEPTS

• **Security and privacy → Authentication**;

KEYWORDS

Measurement, Rate-limiting, Authentication

ACM Reference Format:
Bo Lu, Xiaokuan Zhang, Ziman Ling, Yinqian Zhang, and Zhiqiang Lin. 2018. A Measurement Study of Authentication Rate-Limiting Mechanisms of Modern Websites. In *2018 Annual Computer Security Applications Conference (ACSAC '18), December 3–7, 2018, San Juan, PR, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3274694.3274714

*Co-first authors.

ACSAC '18, December 3–7, 2018, San Juan, PR, USA

ACM ISBN 978-1-4503-6569-7/18/12…$15.00
https://doi.org/10.1145/3274694.3274714

1 INTRODUCTION

Text passwords are strings of human-readable characters that are used as a primary means for user authentication on modern computer systems. Previous studies on password attacks are typically classified into offline attacks and online attacks. Offline attacks assume the adversary's possession of a cryptographically protected password database (*e.g.*, files that store the MD5 hash values of users' passwords), which can be queried arbitrarily to guess the password of any targeted users. In the past decades, various previous work has focused on the security of human-chosen passwords in offline password guessing attacks [21, 24, 26, 30, 32, 38, 44]. These studies have demonstrated more efficient ways to crack hashed passwords than simply brute-forcing. Recognizing the tension between the passwords' memorability and their resilience to password guessing, these research studies have questioned the efficacy of using text passwords for human authentication, and urged the development of alternative authentication methods, such as graphic passwords [14] and biometric-based authentication [20]. Nevertheless, because these alternative authentication methods all have certain limitations in practical applications, to date, text passwords remain the most widely used authentication methods for websites on the Internet [5].

Authentication systems with Internet accesses are vulnerable to online attacks. In contrast to offline password attacks, online password attacks assume lockout or throttling mechanisms implemented in the authentication system to prevent frequent, unsuccessful login attempts, but aim to guess the correct passwords within a number of attempts. Unfortunately, the threats of online password guessing have been underestimated. As user accounts are typically locked out after a few unsuccessful login attempts, a common belief is that online password cracking have been mitigated because of the huge password guessing space and the limited number of allowed guessing attempts. Nevertheless, recent studies have revealed that with some auxiliary information of the targeted account (e.g., the user's personal information, her previously used passwords, or passwords used in other systems) the adversary may successfully guess the human-chosen passwords within a small number of attempts [13, 23, 25, 42, 43, 47].

Particularly, Wang *et al.* [43] have very recently developed efficient guessing algorithms for targeted online password attacks

with the help of auxiliary information of the targeted accounts. The results were astonishing: Within 100 guesses, the algorithms could successfully crack 73% of the normal users' accounts and 32% of the security-savvy users' accounts (when evaluated against leaked password databases). These results have challenged NIST's Digital Authentication Guideline [17] which suggested that "Online attacks where the attacker attempts to log in by guessing the password can be readily addressed by throttling the rate of login attempts permitted."[1] In light of these results, it becomes more important to understand if modern authentication systems have implemented proper countermeasures against online password attacks by limiting the number of incorrect login attempts.

In this paper, we empirically evaluate the feasibility of conducting automated, targeted online password guessing attacks against popular websites. More specifically, instead of assessing the guessability of the user-selected passwords, our study focuses on the mechanisms implemented by these websites to thwart frequent unsuccessful login attempts. We call such mechanisms *authentication rate-limiting mechanisms*, which include both *account lockout* and *login throttling*. Account lockout is a widely adopted mechanism by websites or other authentication systems to defend against online password cracking attacks. If multiple unsuccessful login attempts against an account have been detected within a short period, the account will be "locked" and no one—not even the user with correct passwords—can log in the account. Login throttling usually refers to mechanisms to throttle high-rate login attempts, for example, by using CAPTCHA [41] or temporary (*e.g.*, 5 minutes) account lockouts.

This paper aims to investigate the rate-limiting mechanisms as a whole using a black-box approach. Specifically, without the source code of the websites, we hope to answer the following three questions: (1) How do websites treat multiple login attempts as authentication requests from the same user/attacker? (2) How many times do websites allow the same user/attacker to log in unsuccessfully without being blocked or throttled? (3) How is it correlated with the speed of the login attempts? In doing so, we are able to understand the susceptibility of the website to online password cracking attacks. To the best of our knowledge, due to the closed-source nature of password authentication implementation on major websites, these research questions have not been addressed in previous work.

More concretely, our method first builds a parameterized model for the implemented rate-limiting mechanisms, by considering several login variables (*e.g.*, client IP addresses, browser cookies, the targeted user accounts, *etc.*), and then conducts a sequence of black-box tests to identify the parameters of the model. We applied the approach to study 182 websites in the Alexa Top 500 list in the U.S., which already represents our best effort. The remaining 318 websites cannot be analyzed for various reasons. For example, some do not provide user authentication, some require foreign phone numbers, some reuse an identity management system for which we have analyzed already (multiple domains of `google.com`). Other reasons are explained in Sec. 4.2.

[1] This remark has been hence removed from the document after the publication of [43].

The results of our measurement study are surprising: 131 websites (72%) of all websites allow frequent, unsuccessful login attempts without account lockout or login throttling. Among these 131 websites, some of these websites force the adversary to lower the login frequency to circumvent the rate-limiting enforcement; however, we show in our measurements that the adversary could at least test 84 passwords per day on these websites and, on 48% of these websites, more than 10800 times per day. These websites allow the adversary to perform effective online password guessing as suggested in previous work [43]. 19 out of the 131 websites allow unlimited login attempts if the adversary changes his IP address. Such a restriction, however, is not effective as the adversary can easily rent cloud servers and utilize a large number of cloud IP addresses to conduct the attacks, similar to our experiment setting in this paper. One of these 19 websites allows unlimited login attempts if the adversary cleanses the cookies from the browsers, because these websites track the user using cookies embedded in the browsers. These results suggest that automated, targeted online password guessing is feasible in practice. For the remaining 51 websites (28%) that have enforced rate limiting, our study suggests that they are not absolutely secure: 28 of these websites will block all login requests to the targeted accounts, including those from the legitimate users, when the account lockout is triggered by the attacker. Unfortunately, this design choice effectively enables authentication denial-of-service attacks, in which an attacker could simply send large amounts of login requests to the targeted accounts with incorrect passwords.

Contributions. This paper contributes to the study of password security in the following aspects:

- The paper proposes a novel approach to modeling the rate-limiting implementations of websites, enabling the measurement of their resilience to online password attacks in a black-box manner.
- The paper presents a systematic measurement study on a subset of Alexa Top 500 websites, showing that most of these websites have implemented a rate-limiting mechanisms that can be circumvented by attackers.
- Combined with recent developments of password guessing algorithms using account auxiliary information, the study confirms that online password attacks are practical threats to websites.

Ethical guidelines. This paper involves conducting experiments on public websites to reverse-engineer their implemented mechanisms to thwart online password cracking. However, all login attempts were against user accounts created by ourselves. We have not performed any experiment to crack passwords of other user accounts. Due to this reason, we did not perform evaluations on the websites on which we could not create accounts. Moreover, throughout our experiments, the maximum rate with which the login requests were sent to any specific website was about 1 request per second. In most cases, the actual rate was much lower. Therefore, the impact of our experiments on the performance of these websites would not cause any observable performance degradation to the web servers. We also tried our best to notify the susceptible websites listed in the paper. The procedure of responsible disclosure is discussed in Sec. 6.

Roadmap. The remainder of the paper is outlined as follows. In Sec. 2, we introduce background and summarize related work. In Sec. 3, we detail the proposed black-box tests and in Sec. 4 we present the method in which we setup our measurement study. Sec. 5 performs the analysis of the results. We discuss implications and limitations of our study in Sec. 6 and conclude the paper in Sec. 7.

2 BACKGROUND AND RELATED WORK

In this section, we introduce some background knowledge of the paper and also discuss work related to ours.

2.1 Online Password Guessing Attacks

The security of password authentication has been a topic of research for decades. Previous studies have studied both offline password attacks and online password attacks. Our study is closely related to the research of online password attacks. Specifically, Zhang *et al.* [47] investigated whether modern password expiration is an effective security mechanism against password guessing attacks. Using a new algorithm for searching new passwords from the old ones, their work has shown that 41% accounts can be broken within 3 seconds in an offline setting and 17% accounts within 5 attempts in an online setting. Mazurek *et al.* [25] studied password guessability and found that the strength of a user's password is highly related to her background, such as the area of study, gender,*etc.* Bauman *et al.* [4] studied how website stored user's password, and they found a large number of websites including 11 Alexa's top 500 websites stored plaintext password. Das *et al.* [13] analyzed cross-site password security. Their study found that over 43% of users use the same passwords between different websites. They developed a cross-site password-guessing algorithm, which is able to crack 30% user accounts within 100 attempts. Li *et al.* studied how users embedded their personal information into their passwords [23] and developed Personal-Probabilistic Context-Free Grammars (Personal-PCFG) to crack passwords based on personalized guesses. Similarly, recent work by Wang *et al.* evaluated the password reuse across different services [42]. They examined 107 services and over 61 million passwords, and found that 38% users use the same password across sites.

Our study is motivated by Wang *et al.* who investigated *targeted online guessing*, where the attacker leverages his knowledge of a user's password from a sister website and some personally identifiable information (PII) to conduct online password guessing [43]. Their results suggest that within 100 guesses 73% of the normal users' accounts and 32% of the security-savvy users' accounts can be broken. However, similar to all other previous work listed here, their experiments were evaluated using leaked password databases (or expired files from the authors' organization). The implementation of rate-limiting mechanisms on popular websites was not studied in the paper. In fact, the lack of rate-limiting mechanisms can easily lead to the password brute-force attacks, as demonstrated by AutoForge [49], which discovered many of the remote servers of mobile apps were subject to such attacks.

2.2 Defenses against Online Password Attacks

Account lockout has been a primary defense against online password cracking. Early computer systems used to limit unsuccessful login attempts to a very small number, such as 3 times. Early back in 2003, Brostoff and Sasse [7] argues that by increasing the limit from 3 times to 10 times, the usability of the authentication system can be improved. Similar arguments were also made by Renaud *et al.* [33]. Most modern computer systems implement a higher threshold for account lockout. It is recommended by NIST's Digital Identity Guidelines [17] that "the verifier SHALL limit consecutive failed authentication attempts on a single account to no more than 100". In 2010, Bonneau and Preibusch [6] conducted a simple test on 150 randomly selected websites and found 126 of them allowed more than 100 repeated login attempts. Our study systematically extends Bonneau and Preibusch [6]. While their study only analyzed the number of unsuccessful logins that would trigger rate-limiting, our method also considers and measures the effects of login intervals and other factors (such as changing IP addresses and cleansing browser cookies). Moreover, instead of randomly selecting 150 websites [6], in our study, we evaluated all (182) websites on the Alexa Top 500 list that could be studied. The results of our study represent the status quo of most high-profile websites. The negative side effect of account lockout has been discussed in a few prior studies [15]. For example, Florencio *et al.* [15] pointed out the challenges of designing a proper rate-limiting policy: usability and denial of service. Our study offers one additional piece of evidence that advocates further research of password security to resist online password cracking.

Beside account lockout, a popular way to implement rate-limiting is by using CAPTCHA [2, 31, 39, 41], which presents a hard artificial intelligence problem that a computer program cannot solve and forces the human user to manually intervene the authentication process. CAPTCHA is primarily designed to prevent automated password guessing attacks using computer programs. The usability of CAPTCHA has been an issue. Moreover, various studies have demonstrated that most CAPTCHA systems can be bypassed [3, 8–10, 12, 16, 22, 28, 29, 36, 48].

2.3 Offline Password Attacks

Offline password attacks assume the adversary's possession of a cryptographically protected password database (*e.g.*, MD5 hash files), which can be queried arbitrarily to guess the password of any specific users. Therefore, research on offline password security typically aims to develop more efficient tools to crack hashed password files than brute-force methods [19, 21, 24, 26, 30, 32, 38, 40, 44]. For instance, Weir *et al.* proposed a new way to generate password candidates using probabilistic context-free grammar (PCFG) and show that they can be used to crack password files more efficiently [44]. Ur *et al.* studied various password cracking approaches used by researchers and professionals [38]. A summary of these studies can be found in a survey paper by Han *et al.* [18].

3 METHODOLOGY

To understand how websites block online password cracking attacks, we derived a sequence of black-box tests to explore (1) how websites treat multiple login attempts as authentication requests

Algorithm 1: *Lockout Threshold* Tests

```
Input:
    Accounts = [] ;                          /* initialize a list of accounts */
    IPs = [] ;                               /* initialize a list of ip addresses */
    Intervals = [] ;                         /* A list of itvl to test */
Output:
    out ;                                    /* A list of (itvl, thrsh) */
begin
    out = [];
    repeat
        itvl = Intervals.next();
        Account = Accounts.next();
        IP = IPs.next();
        Cookie.clear();
        counter = 0;
        repeat
            ret = UnitTest(Account, IP, Cookie);
            if ret = L then
                break;
            end
            wait for itvl seconds;
            counter = counter + 1;
        until counter > 100;
        out = out + (itvl, counter) ;                /* store the result */
        if counter > 100 then
            break ;                          /* no need to test the rest of itvl  */
        end
    until Intervals.next();
    return out
end
```

Algorithm 2: *Counting Mechanism* Tests

```
Input:
    Accounts = [] ;                          /* initialize a list of accounts */
    IPs = [] ;                               /* initialize a list of ip addresses */
    Threshold = G(itvl) ;                    /* itvl = 8s, a small value */
Output:
    Out ;          /* A list of (acct, cip, cookie) that create clean states */
begin
    out = [];
    Create a clean state with new account, ip, and delete cookie;
    for cookie = 0 to 1 do
        for cip = 0 to 1 do
            for acct = 0 to 1 do
                if cookie == 1 then
                    Cookie.clear()
                end
                if cip == 1 then
                    IP = IPs.next()
                end
                if acct == 1 then
                    Account = Accounts.next()
                end
                counter = 0;
                repeat
                    ret = UnitTest(Account, IP, Cookie);
                    if ret = L then
                        break;
                    end
                    wait for itvl seconds;
                    counter = counter + 1;
                until counter == Threshold;
                if counter == Threshold then
                    out = out + (acct, cip, cookie);
                end
            end
        end
    end
    return out
end
```

from the same user (and thus increment a counter to track the history of logins), and (2) how many times the websites allow the same user to send frequent login attempts without account lockout or login throttling. In the context of this paper, we do not distinguish lockout and throttling as they both thwart continuous login attempts from an automated online password cracker. We define:

- *Counting Mechanism*: The mechanism with which a website identifies multiple continuous login attempts from the same user.
- *Lockout Threshold*: The maximum number of invalid login attempts from the same user that can occur before the account is locked out or throttled.
- *Transition Interval*: The minimum interval between consecutive password guesses that allows the adversary to continuously login without blocking.

Therefore, the primary research goal of this paper is to investigate a website's *Lockout Threshold*, *Counting Mechanism*, and *Transition Interval*.

3.1 Modeling *Lockout Threshold* and *Counting Mechanism*

To explore a website's *Lockout Threshold* and *Counting Mechanism*, we propose a black-box approach, in which we first construct a hypothesis of the implemented *Lockout Threshold* and *Counting Mechanism*, and then test the hypothesis empirically.

A model of *Lockout Threshold*. We model a *Lockout Threshold* as a function $G(itvl) \rightarrow \mathbb{N}$, where *itvl* is the time interval between two consecutive logins and $\mathbb{N}$ is the set of all natural numbers. This model abstracts away complex details of the websites' implementation of the *timeout* mechanisms. For example, a simple implementation of the *timeout* setting is to maintain a timer on every failed login history and discard the history when the timer expires. A more complex implementation choice is using an *exponential moving average* of the history, which will place a heavier weight on the more recent attempts and gradually "forget" the old records. Due to the large space of design choices, we chose to adopt a black-box approach and abstract away the concept of the timeout and only use the *itvl* between logins to represent the internal implementation. Although our model simplifies the internal implementation details, this level of abstraction is sufficient for our intended purposes.

A model of *Counting Mechanism*. We assume a website's *Counting Mechanism* is determined only by three factors (dubbed *lockout factors*): (1) Is the targeted account the same as previous ones in a failed login attempt? (2) Is the client IP address the same as previous ones that have attempted to log in but failed? (3) Does the browser cookie that tracks client information tells the website that the user has attempted to log in (but failed) before?

We use three variables *acct*, *cip*, and *cookie* to indicate whether these three *lockout factors* have been *changed*—whether they are the same as a *previously seen* login attempt. We let $acct = 0$ if the account has been targeted by previously seen logins; $cip = 0$ if the client IP has been used to log in; $cookie = 0$ if the cookie sent together with the login request indicate the user has recently attempted to log in. Otherwise, $acct = 1$, $cip = 1$, and $cookie = 1$.

Therefore, we model *Counting Mechanism* as $F(acct, cip, cookie) \rightarrow \{0, 1\}$. $F = 1$ means the *Counting Mechanism* has recognized the user as one that has previously attempted to log in but failed, and thus a counter associated with that user will increment by 1. $F = 0$

means the *Counting Mechanism* has not recognized the user, and hence this login attempts will start from a *clean state* and create a new counter to track the user.

We assume the *Counting Mechanism* that a website implements does not change for different targeted accounts, different client IP address, or different cookie values. We further assume $F(1, 1, 1) = 0$; that is when the user tries to log in to a different account, from a different client IP address, and without any tracking cookies, the website will not recognize the user.

The definition of "previously seen" requires more attention. A website cannot store the previous login attempts forever and compare every failed authenticate request with the old record. A timeout mechanism must be implemented for the websites to forget about the history. We abstract this timeout mechanism using a fourth variable *itvl*, which describes the interval between two consecutive logins.

3.2 Black-box Tests

We treat websites' implementation of defenses against online password guessing attacks (*i.e.*, *Lockout Threshold* and *Counting Mechanism*) as a black box, and conduct two sets of black-box tests, one for *Lockout Threshold* and the other for *Counting Mechanism*, to reverse engineer its implementation. Both these two tests are designed as sequences of unit tests.

Unit tests. Each unit test is provided with three input variables: the tested account of the targeted website, the client IP address to request authentication, and the history of previous login attempts embedded in the browser cookies. When the browser tries to log in with the three input variables, the website will return one of the three values: Success (S), Wrong Password (W) and Locked (L). Different websites may lock out the user by different means. For example, a website may display a CAPTCHA for the user to solve, or show a page indicating that this account has been locked. The output of the unit test reports the return value from the website accordingly. Formally, the unit test is denoted as $UnitTest(Account, IP, Cookie) \rightarrow \{S, W, L\}$.

***Lockout Threshold* tests.** To test the correlation between the intervals between two consecutive logins from the same user (*i.e.*, *itvl*) and the maximum number of unsuccessful login attempts allowed by the websites before the user is locked out (*i.e.*, *thrsh*), we conducted the experiments depicted in Algorithm 1. Specifically, the algorithm first initializes a list of accounts to be tested, a list of IP address to be used as the client IP addresses and a list of *itvl* the *Lockout Threshold* test will examine (line 1-3). Then the algorithm repeats until $Intervals.next()$ returns empty (line 25). In each iteration (line 8-24), the algorithm selects a new account (line 9), a new client IP address (line 10), and clears the cookie (line 11). Then it initiates a sequence of $UnitTest()$ with the account and IP address, while keeping the cookies. The interval between two $UnitTest()$ is the underlying *itvl* of the current iteration (line 18). The sequence of $UnitTest()$ continues until it returns L in the test (line 15-17) or the test goes beyond 100 times, a value we believe is large enough to indicate a by-pass of the website's rate-limiting mechanism (line 20). The number of times $UnitTest()$ is called, *counter*, is recorded as the corresponding *thrsh* of *itvl*. As the list of *itvls* is sorted in a non-descending order, if the *thrsh* value for an *itvl* exceeds 100, those for larger *itvl*s are also expected to exceed 100, and thus these tests will be skipped (line 29-31).

***Counting Mechanism* tests.** To test the *Counting Mechanism*, F, we need to determine all 3-tuples of the *lockout factors* that the websites use to identify a user: $\{(acct, cip, cookie) | F(acct, cip, cookie) = 0\}$. In order to determine if $F(acct, cip, cookie) = 0$ for a given 3-tuple, we first conduct the following experiments: We start a unit test with a previously unused account from a different client IP address without any cookie in the browser, and repeat this unit test with the same account and the same IP address but without cleansing the cookies until the unit test returns L. The interval between logins, *itvl*, is a small value (*e.g.*, 8 seconds) that is much shorter than the possible value of the timeout mechanisms implemented in any website. The total number of login attempts are recorded and denoted $G(itvl)$.

Then we run the algorithm illustrated in Algorithm 2 to identify all 3-tuples $(acct, cip, cookie)$ that satisfy $F(acct, cip, cookie) = 0$. Specifically, in Algorithm 2, line 1-2 initialize a list of user accounts and a list of client IP addresses for the tests. The methods $Accounts.next()$ (line 18) and $IPs.next()$ (line 14) retrieve the next element in the lists. Then the algorithm enumerates all 8 (*i.e.*, 2^3) possible combinations of $(acct, cip, cookie)$ (line 8-10) and conducts 8 sequences of $UnitTests()$ (line 11-31). In each test sequence, first of all, the account, IP address and cookie settings for the unit test are selected in accordance with the value of the 3-tuple (line 11-19). Then the unit test is repeated until the number of tests exceeds 100 (line 28), a value that is large enough to indicate a by-pass of the website's rate-limiting mechanism[2], or a value L is returned from the unit test (line 23-25). Algorithm 2 reports all the 3-tuples that the corresponding test sequence can repeat the unit test for $G(itvl)$.

4 A MEASUREMENT STUDY OF RATE LIMITING IMPLEMENTATIONS

In this section, we describe how we setup our measurement study and collect data.

4.1 Experiment Setup

To automate the black-box tests outlined in Sec. 3, we developed a Python script using the Selenium framework [34]. Selenium is a browser automation tool that enables programmatically operating (*e.g.*, click a button to POST a form) the web interface exposed through HTML files and observe the responses from the websites. Selenium provides the options to delete cookies from the browsers. This feature is used to create tests associated with the *cookie* variables. To enable dynamically changing targeted accounts, we manually registered multiple user accounts on each of the tested websites. This account registration step precludes some of the websites from our datasets, which we will elaborate shortly.

To change client IP addresses dynamically, we set up our test environment on Microsoft Azure Cloud [27]. The benefit of using clouds to conduct experiments is the scalable IP pools provided by the cloud provider as well as the large number of VMs that greatly parallel the experiments. More specifically, we set up proxy

[2]The latest version of NIST's Digital Authentication Guideline [17] suggests that "the verifier SHALL limit consecutive failed authentication attempts on a single account to no more than 100".

servers on multiple VMs in Azure using Shadowsocks [35]. The proxy clients are installed on another set of Azure VMs and our lab desktops. Selenium is used to interact with the browser, which visits the target websites through the proxy servers.

The Selenium framework allows us to dynamically adjust the intervals between consecutive login requests. But note that Selenium simulates the complete login process, which includes downloading the login forms, filling in the content of the forms, and uploading the forms using HTTP POST methods. These steps are not avoidable because each login request should be sent together with a random token associated with the login form and created by the website to prevent cross-site request forgery. Therefore, the intervals between consecutive logins are also determined by the latency of the HTTP requests.

4.2 Data Collection

We conducted a measurement study on websites selected from The Alexa Top 500 websites[3] in United States [1]. The experiments were conducted by following the workflow shown in Fig. 1. Specifically, for each website in the Alexa Top 500 list, we *first* registered an account with the website manually. If for any reason we could not complete this step (*e.g.*, did not allow account registration), we skipped the remaining steps and moved on to the next website. *Second*, we examined the login page of the website to find elements of username fields, password fields, and the login button by inspecting the code; these elements were then used to customize the Python scripts to operation Selenium. We then performed the *Lockout Threshold* test with only one interval (*i.e.*, 8 seconds) in the *Intervals* array of the input. If the corresponding *thrsh* in the output is greater than 100, it means the website did not implement a proper rate-limiting mechanism. We did not perform the *Counting Mechanism* test on these websites, because the test will certainly return all 8 combinations in the output as they will all reach 100 login attempts.

For all other websites, we manually inspected the lockout page (*e.g.*, the web page that indicates the account has been locked out or shows a CAPTCHA for the user to solve) to identify elements that only exist in the lockout page (*e.g.*, A unique URL) and customize the Python scripts to recognize this page. After that, we registered 7 more accounts and performed the *Counting Mechanism* test as detailed in Sec. 3, and the *Lockout Threshold* test with $itvl = [16s, 32s, 64s, 128s, 256s, 512s, 1024s]$.

We studied all Alexa Top 500 websites. However, 318 websites had to be excluded from the tests: ❶ 24 of them contained sexual or illegal content, such as `pornhub.com` and `thepiratebay.org`; we intentionally removed these websites from our dataset. ❷ 50 websites either used single sign-on services from other websites (*e.g.*, using Google account) for authentication, or it would redirect users to another Alexa top 500 website (*e.g.*, `pinimg.com` redirected users to `pinterest.com`); these websites cannot be examined independently. ❸ 69 websites did not have user profiles (*e.g.*, `ca.gov`). ❹ 104 websites required a mobile phone number, credit card number, social security number, or invitation code, which makes it impossible for us to register multiple accounts. ❺ We encountered various technical issues when testing the remaining 71 websites, and we could not perform our analysis on these websites. We will discuss the limitation of our cloud experiments in detail in Sec. 6. Although we only eventually tested 182 websites among the Alexa Top 500 websites, due to the nature of the *Lockout Threshold* tests and *Counting Mechanism* tests, data collection for these websites took us 5 months and several thousand US dollars in cloud usage[4].

[3] Accessed on Dec 21, 2017.

5 EVALUATION AND ANALYSIS

In this section, we analyze the data collected in the measurement study and discuss the security implication of the results. In Sec. 5.1, we report our experiment results of the *Counting Mechanism* and *Lockout Threshold* tests directly; in Sec. 5.2, we interpret the experiment results in the context of online password attacks.

5.1 Data Analysis

According to Fig. 1, for each of the 182 websites that we tested, we first conducted a *Lockout Threshold* test with $itvl = 8s$, which serves as a preliminary test to identify websites that apparently do not enforce rate-limiting. If we observe $thrsh > 100$ in this test, the website is included in dataset A. Otherwise, if $thrsh \leq 100$, the website is included in dataset B. Of the 182 websites, 63 websites were in dataset A, and 119 websites were in dataset B. Websites in the two datasets are processed differently. On each of the 63 websites in dataset A, we only performed *Lockout Threshold* tests with *itvl* less than 8 seconds (because for larger *itvl*s clearly $thrsh > 100$). On each of the 119 websites in dataset B, *Lockout Threshold* tests with $itvl = [16s, 32s, 64s, 128s, 256s, 512s, 1024s]$ and *Counting Mechanism* tests were conducted. The experiment results are detailed as follows.

***Lockout Threshold* Tests.** The lockout thresholds of all 182 websites in both dataset A and B are examined. In Fig. 2a, the x-axis is the lockout threshold, *thrsh*, and the y-axis is the cumulative distribution function (CDF) of the total number of websites on which the *Lockout Threshold* tests could reach the corresponding *thrsh* when the websites were probed with a specific interval, *itvl*. From Fig. 2a, we can see that when $itvl = 8s$, the *thrsh* of about 63% of the websites are less than 20 (*i.e.*, the intersection of the vertical line for $thrsh = 20$ and the higher blue line in the figure is 0.63 on the y-axis). When the interval between consecutive login attempts, *itvl*, increases, the maximum number of logins, *thrsh*, allowed before the user is blocked or throttled will also increase. For example, when $itvl = 128s$, the *thrsh* of 50% of the websites are more than 20 (*i.e.*, the intersection of the horizontal line for $CDF = 50\%$ and the purple line in the figure is 20 on the x-axis). The *thrsh* of more than 46% of the websites are more than 100—the maximum *thrsh* we tested (*i.e.*, the vertical line for $thrsh = 100$ and the purple line in the figure intersects at about 0.54 on the y-axis). When $itvl = 1024s$, which is about 17 minutes, the *thrsh* of roughly 60% of websites are more than 100. It means that 60% websites do not block a login rate of 3.52 (=3600/1024) times per hour.

Fig. 2b and Fig. 2c illustrate in more details the distribution of *thrsh* among websites when $itvl = 8s$ and $itvl = 1024s$, respectively. The *Lockout Threshold* of these websites increases as *itvl* increases: When $itvl = 8s$, nearly 80 websites limit login attempts to 10; in

[4] Close to the end of our data collection, we switched to Cloudlab to reduce the monetary cost of the experiments.

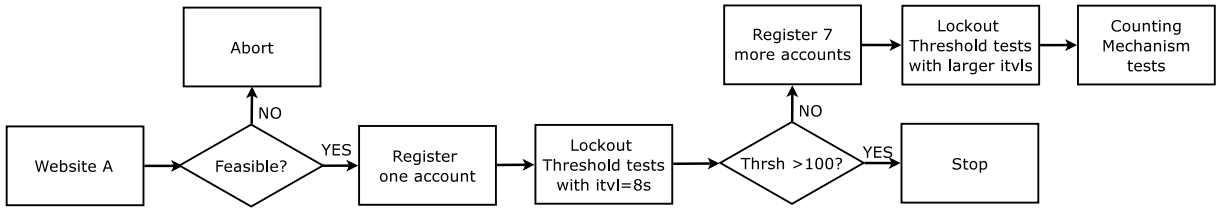
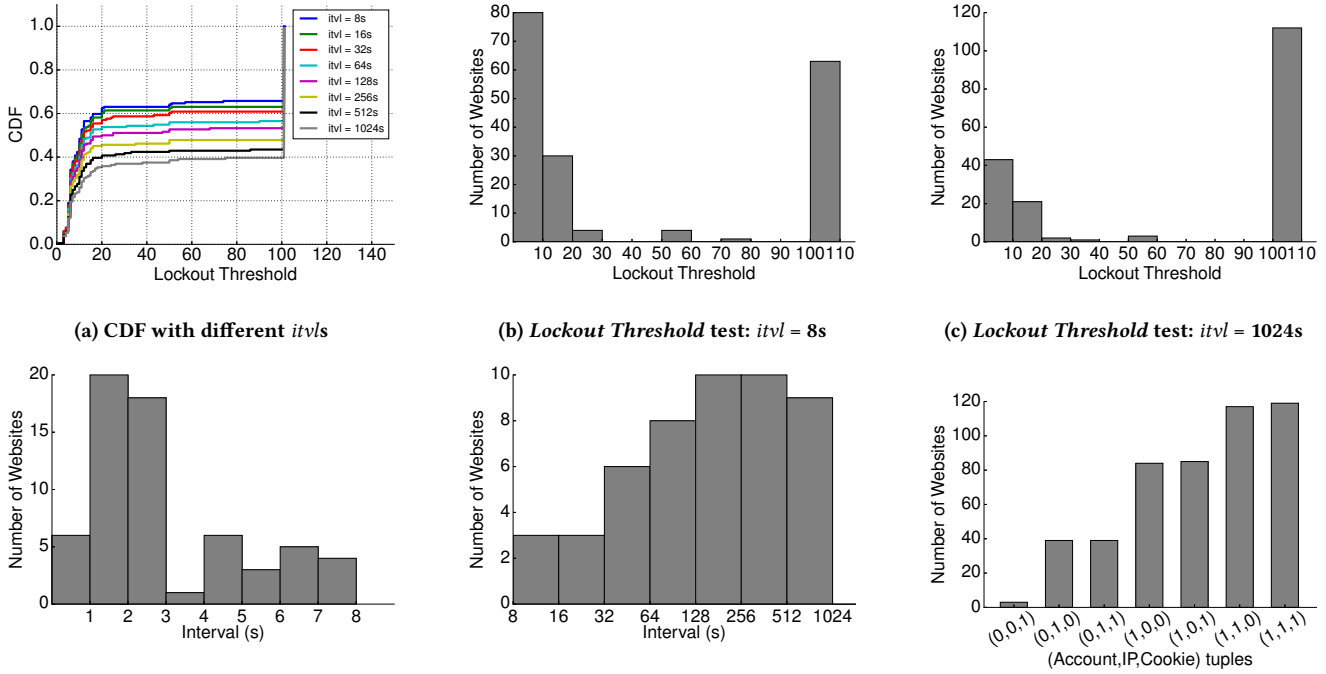


Figure 1: Workflow of the data collection process.



(a) CDF with different *itvl*s

(b) *Lockout Threshold* test: *itvl* = 8s

(c) *Lockout Threshold* test: *itvl* = 1024s

(d) *Transition Interval* of the websites in dataset A

(e) *Transition Interval* of the websites in dataset B (*thrsh* ≥ 100)

(f) *Counting Mechanism* tests

Figure 2: Results of data analysis. Fig. 2a, Fig. 2c, and Fig. 2b show the results related to *Lockout Threshold*; Fig. 2d and Fig. 2e show results related to *Transition Interval*; Fig. 2f shows analysis of *Counting Mechanism*.

contrast, when *itvl* = 1024*s*, only 43 websites still limit the login attempts to 10. Moreover, when *itvl* is set to 1024 (as shown in Fig. 2c), the adversary could reach 100 login attempts without encountering rate-limiting on 112 websites, 49 more than that shown in Fig. 2b. We particularly studied these 49 websites that are no longer blocked to find out at which login interval, *itvl*, did the websites stop blocking repeated logins (*i.e.*, *Transition Intervals* of the websites). That is, we hope to determine the maximum allowed login rate of these websites. The *Transition Intervals* of these 49 websites are shown in Fig. 2e. We can see from the figure that 3 websites' *Transition Intervals* are between 8 to 16 seconds; 8 websites have *Transition Intervals* between 64 to 128 seconds; only 9 websites have *Transition Intervals* larger than 512 seconds.

To determine the *Transition Intervals* of the 63 websites in dataset A, we altered our experiment code to repeatedly authenticate with the website without any intentional delay. It is worth noting that for most of these websites the *Transition Intervals* are above zero due to network latency, rather than rate-limiting. However, some of these websites (*e.g.*, `stackoverflow.com`, `steampowered.com`, `theguardian.com`, and `target.com`) do present CAPTCHA or lockout account to thwart password guessing when the intervals between failed logins are too small. Therefore, we had to intentionally slow down the login requests to determine the *Transition Intervals* of these websites. The result is shown in Fig. 2d. From the figure, we can see that 20 of the 63 websites (32%) allows a login interval of 1 to 2 seconds, 6 websites even allow an interval of less than 1 second.

***Counting Mechanism* Tests.** For the 119 websites in dataset B which indicate that the account is locked out or shows a CAPTCHA, we performed the *Counting Mechanism* tests to determine how the websites identify the authentication requests are from the same users (as described in Algorithm 2). Fig. 2f illustrates the number

of websites on which we can create a clean state by altering one of three variables in the (*acct*, *cip*, *cookie*) tuple. Recall (1, 1, 1) means all three factors are changed between logins; and (0, 0, 0) means no changes are made. Note that for a given tuple (*e.g.* (1, 1, 0)), if one of its subset (*e.g.* $(1, 0, 0) \subseteq (1, 1, 0)$) can succeed, it will also succeed.

From Fig. 2f, we find that if we restrict ourselves to only changing one variable, changing *acct* (*i.e.*, (1, 0, 0)) is the most effective method, as it works for 84 out of 119 websites. This means 84 websites accumulate a counter based on which accounts the user attempts to log in. When we only change the client IP address, *cip*, the experiments could create clean states on 39 websites (*i.e.*,(0, 1, 0)). This means these 39 websites track users by their IP addresses rather than which accounts they attempt to log in. Surprisingly, we found that for 3 websites (*i.e.*,(0, 0, 1)), only cleansing the browser cookies would be sufficient to reach a clean state and circumvent the rate-limiting mechanism. Note there are 3 websites on which either changing IP or cleansing cookies would work. This can be seen as (0, 1, 1) corresponds to 39 websites in Fig. 2f.

Take-aways. The data analysis enables the inference of rate-limiting implementation of the investigated websites. First, the *Lockout Threshold* tests examine whether rate-limiting is at all enforced and, if so, what is the maximum login rate for each of them that would not trigger rate-limiting. Our finding suggests that with sufficiently large *itvl* (*i.e.*, low login rate), most websites would allow unlimited failed login attempts (*e.g.*, 112 or 62% websites when $itvl = 1024s$). Moreover, it is also surprising to know that *Transition Interval* of most websites is very low (*e.g.*, 63 or 35% websites have a *Transition Interval* lower than 8*s*), which means an adversary could conduct high frequency password guessing attacks against these websites. Second, the *Counting Mechanism* tests explore how rate-limiting is enforced and how an attacker might circumvent the enforcement. Among the 119 websites in dataset B that clearly enforce some level of rate-limiting, some (*i.e.*, 84 websites) track user login by account, some (*i.e.*, 39 websites) by client IP addresses, some (*i.e.*, 3 websites) by cookie values, and some by a combination of these factors. In a targeted password guessing attack, as the attacker could change his IP address or cleanse browser cookies, the rate-limiting mechanisms of 39 websites can be easily bypassed.

5.2 Security Analysis

According to the data analysis, we found that the examined websites had implemented different levels of resilience to online password cracking attacks. For some websites, the adversary could bypass the counting mechanisms implemented by the websites by changing his request intervals, or IP addresses, or browser states (*e.g.*, cookies). These websites allow the adversary to test an unlimited number of passwords online without being locked out or throttled. We call these websites *susceptible websites*. Other websites have stronger defenses against online password cracking attacks, which we call the *low-risk websites*.

Susceptible websites. Specifically, in our measurement study, a susceptible website is a website that allows the adversary to perform 100 login attempts on one account without being locked out or throttled with $itvl \leq 1024s$ (possibly by changing their IP addresses or cleansing browser cookies). 131 out of the 182 websites belong to this category (listed in Table 2). Among all the susceptible websites in our study, 112 of them permit repeated logins with $itvl = 1024s$; another 19 of them that are not able to reach $thrsh = 100$ can do so successfully with the help of changing IP address. Although websites requiring the adversary to change IP address for attacks demand additional efforts, these websites are still susceptible to online password cracking attacks, as the adversary could easily use cloud IP address, Tor, or VPN services to bypass this enforcement.

Interval (seconds)	Attempts per Minute	Attempts per Hour	Attempts per Day
1	60	3600	86400
4	15	900	21600
16	3.75	225	5400
64	0.94	56.25	1350
256	0.23	14.06	337.50
1024	0.06	3.52	84.38

Table 1: Converting *Transition Interval* to estimated numbers of login attempts per minute, per hour, and per day.

Interestingly, many high profile websites, such as `google`, `reddit`, `facebook`, `twitter`, and `Instagram` all belong to this category. Many financial related websites, such as E-Commerce and banking websites, are also susceptible to online password attacks, including `amazon`, `walmart`, `bestbuy`, `booking.com`, and `coinbase.com`.

One important question to answer for these susceptible websites is how fast could the online password cracking be performed. In Table 1, we convert the minimum interval allowed to bypass the counting mechanism (*i.e.*, *Transition Interval*) to the number of attempts that could be made by an adversary within a minute, an hour, and a day. Particularly, if the login interval is 1 second, within a minute, the adversary can attempt to log in 60 times, and 86400 times in a day. When the requested login interval is 1024 seconds, within a single day, the adversary can send 85 login attempts to the targeted website. This is still significantly more than the NIST suggested 100 times for a 30-day period.

We further estimated how many failed login attempts are allowed per day on the 112 websites that are able to reach $thrsh = 100$ with a single IP, and the result is shown in Fig. 3. From the figure, we can learn that there are 103 websites allow at least 168 login attempts on one account per day; 63 websites allow more than 10800 login attempts on single account within 24 hours; 6 websites even permit more than 86400 attempts on one account within a day.

One way we imagine that can greatly speed up the targeted online password guessing is to conduct the attack in parallel from multiple IP addresses. To confirm that we can perform online password attacks against the same account on the same website in parallel from different machines, we conducted a parallel cracking experiment. In this test, we use N machines ($N = 1, \cdots, 5$) at the same time to log in a website, `archive.org`, whose *Transition Interval* is below 1 second. We recorded the timestamp for each unsuccessful attempt on every machine, and calculated the total time for reaching a certain number of total login attempts, *i.e.*, 100, 300, 500. The results are shown in Fig. 4. From the figure, we see that the time needed to make N guesses is proportional to N. Moreover, to perform online password guessing 500 times, it takes

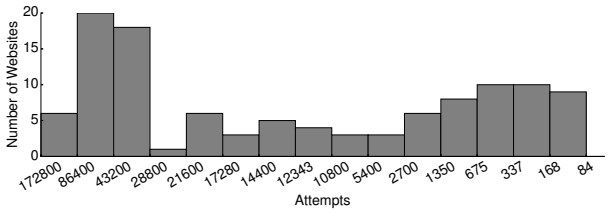


Figure 3: Statistics of susceptible websites: the height of the bars shows the estimated range of the maximum number of failed login attempts per day that can be tried on the websites without being blocked.

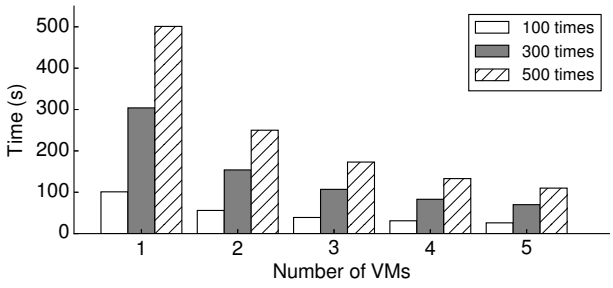


Figure 4: Multiple VMs performing online password guessing in parallel. The height of the bars shows the time needed to send the specified number of requests in total with N VMs. The website under examination is `archive.org`.

the adversary 501 seconds when using one machine, 250 seconds with 2 machines, 173 seconds with 3 machines, 133 seconds with 4 machines, and 110 seconds with 5 machines. This means the attack can be performed in parallel and almost scale linearly.

Low-risk websites. Specifically, in our measurement study, we define a low-risk website as a website that prevents, by implementing effective rate-limiting mechanisms, an adversary to consecutively crack the same account with $itvl \leq 1024s$ even though he is capable of changing the IP address or cleansing browser cookies. 51 out of the 182 websites belong to this category (listed in Table 2). There are two types of low-risk websites. First, 23 of them use CAPTCHA to throttle the login speed. When the user account is locked out, further login attempts will be presented a CAPTCHA; if the CAPTCHA is successfully circumvented, the user is allowed to continue login (*e.g.*., typically once). But if this login fails again, another CAPTCHA will be presented at the time of the next login. We believe such implementation effectively blocks online password attacks while preserve sufficient usability to legitimate users.

The second type of low-risk websites lock out the user account without presenting a CAPTCHA. If so happens, subsequent logins with correct passwords will also be blocked. The legitimate user of the account has to go through a series of efforts to recover the account, such as resetting the password or calling a customer representative. Among the 51 low-risk websites, 28 websites are of this type. We believe these websites enable *Authentication Denial-of-Service Attacks*. An adversary could trigger an account lockout by repeatedly attempting to login the targeted account with wrong passwords from any IP address. When the legitimate user attempts to log in, no matter which IP address is used, her requests will be denied. Such websites include some well-known websites such as `espn.com`, `adidas.com`, `barnesandnoble.com`, `ibm.com`, *etc.* Although the legitimate user can be unlocked after a certain period or unlock herself via certain two-factor authentication mechanisms (*e.g.*, phone calls, text message, emails), persistent authentication DoS attacks might repeatedly lockout the account and bring significant hurdles to the authentication process of the legitimate users. This type of Denial-of-Service attacks have been previously mentioned in the literature [15], and our study shows that they still exist in popular websites.

Take-aways. Although all websites face the threat of online password guessing attacks, some websites are more susceptible to them. Our criteria of susceptibility is whether the attacker can achieve an attack rate higher than 85 attempts per day (*i.e.*, $itvl = 1024s$) without triggering rate-limiting, regardless of the cost of the attacks (*e.g.*, the need to change IP addresses). Under this definition, 131 out of the 182 investigated websites are susceptible to online password guessing attacks. Most websites we evaluated notably allow a much higher login rate (as shown in Fig. 3). Our study also suggest conducting online password guessing attacks from multiple machines in parallel may drastically increase the attack rate on some websites.

The remaining 51 websites are not susceptible under this definition. However, because we were not able to extend our measurement to include *Lockout Threshold* tests with $itvl > 1024s$, it is possible some of them may allow a lower login rate (less than 85 attempts a day) without enforcing rate-limiting, which nevertheless is still high enough to endanger some user accounts with weak passwords. Moreover, while our analysis considers these websites low-risk, it is worth noting over half of the 51 websites enable a form of Denial-of-Service attacks, which is particularly threatening, since attackers may lock the user account on these websites using wrong passwords from any IP address.

5.3 Interesting Observations

During our analysis, we found some websites implement mechanisms that make online password cracking more difficult, although it may not be the intention of the website administrators. We list some of these mechanisms here:

Indifferentiable lockout state. Some websites, such as `genius.com`, never explicitly inform the users that their accounts have been locked out. After several unsuccessful login attempts, the users cannot even log in with correct passwords. Because there is no indication of rate limiting or login throttling, we could not conduct analysis on these websites.

Advanced attack detection. Some websites, *e.g.*, `soundcloud.com`, are able to detect automated authentication requests sent from Selenium. After we have sent login requests to `soundcloud.com` for more than 100 times, with some probability, CAPTCHA will be displayed together with a note to the user suggesting that a robot has been detected. We only observed one such website in our dataset. Unfortunately, we could not determine how the detection scheme

Risk-level	Websites
Susceptible (131)	github.com / spotify.com / dailymail.co.uk / theguardian.com / npr.org / goodreads.com / wunderground.com / politico.com go.com / cnn.com / stackoverflow.com / buzzfeed.com / weather.com / forbes.com / businessinsider.com / steampowered.com nordstrom.com / bleachereport.com / latimes.com / shutterfly.com / dingit.tv / streamable.com / investopedia.com giphy.com / ultimate-guitar.com / speedtest.net / khanacademy.org / gyazo.com / urbanoutfitters.com / staples.com jcrew.com / telegraph.co.uk / independent.co.uk / marketwatch.com / chron.com / biblegateway.com / barstoolsports.com si.com / fanfiction.net / evite.com / rotoworld.com / grubhub.com / forever21.com / pcpartpicker.com / pixnet.net instagram.com / godaddy.com / weebly.com / amazon.com / joinhoney.com / ebay.com / imdb.com / etsy.com / washingtonpost.com foxnews.com / newegg.com / groupon.com / wikihow.com / eventbrite.com / rumble.com / outbrain.com / theatlantic.com trulia.com / worldstarhiphop.com / bandcamp.com / dominos.com / myanimelist.net / tomshardware.com / exoclick.com yandex.ru / woot.com / grammarly.com / colourpop.com / sfgate.com / google.com / facebook.com / reddit.com / pinterest.com roblox.com / glassdoor.com / theverge.com / cbssports.com / battle.net / webmd.com / expedia.com / bedbathandbeyond.com canva.com / booking.com / infusionsoft.com / qualtrics.com / udemy.com / shutterstock.com / pbs.org / collegeboard.org mediafire.com / houzz.com / zillow.com / archive.org / wikia.com / gizmodo.com / lifebuzz.com / realtor.com / squarespace.com nydailynews.com / engadget.com / stubhub.com / livestrong.com / popsugar.com / twitter.com / craiglist.com / coinbase.com adobe.com / slate.com / gamespot.com / meetup.com / walmart.com / bestbuy.com / gamestop.com / evernote.com / feedly.com nytimes.com / cnbc.com / nike.com / nhl.com / target.com / yelp.com / indeed.com / asana.com / vimeo.com / wix.com / vox.com
Low-risk (51)	imgur.com / tumblr.com / bbc.com / tripadvisor.com / dell.com / espn.com / usps.com / slickdeals.net / ups.com / quora.com homedepot.com / cnet.com / intuit.com / costco.com / gap.com / ticketmaster.com / mega.nz / crunchyroll.com / patreon.com walgreens.com / jcpenney.com / seekingalpha.com / zoom.com / ikea.com / barnesandnoble.com / dickssportinggoods.com constantcontact.com / ibm.com / zappos.com / foodnetwork.com / coursehero.com / mobafire.com / toysrus.com / gotomeeting.com arstechnica.com / marriott.com / myfitnesspal.com / pch.com / norton.com / yahoo.com / kickstarter.com / upwork.com macys.com / gamefaq.com / fandango.com / adidas.com / wowhead.com / healthcare.gov / samsung.com / rei.com / asos.com

Table 2: The list of all 182 websites that we have examined. 131 of them belong to the *susceptible* category, which means they allow the attacker to perform at least 100 login attempts on one account without being locked out or throttled; 51 of them belong to the *low-risk* category, which means these websites have implemented effective rate-limiting mechanisms to block the attackers.

was implemented. Detecting bots appears to be an interesting research topic by itself.

5.4 Recommendations

After the analysis, we would like to make the following recommendations to the website developers and operators in developing rate-limiting mechanisms. First and foremost, to reduce the risk, a website must implement a counting mechanism that disregards any information from the client side, such as browser cookies and IP addresses, which can be manipulated by the adversary. The only information the website could rely upon is the account being targeted. All low-risk websites in our study have already implemented this mechanism. Equally importantly, the counter should not be reset too frequently, so that the adversary must use longer intervals between two consecutive login attempts to avoid incrementing the counter.

Secondly, a website must implement a proper rate-limiting mechanism when the same account has reached a threshold. Locking down the account to disallow any login attempt, even for a short period, will make the website vulnerable to authentication DOS attacks. This point has also been mentioned in prior studies [15]. It is also impossible to only allow legitimate logins because in this way the adversary will be able to differentiate the correct passwords from incorrect ones. CAPTCHA is a better solution to keep a balance between availability and security. The website should allow legitimate users to solve a CAPTCHA and continue to log in with the correct password. Nonetheless, as recent studies have shown that CAPTCHAs can be circumvented by using various artificial intelligence techniques, solely relying upon CAPTCHA may only increase the cost of online password cracking attacks but not completely eliminate this attack vector. An alternative solution is to enable two-factor authentication only when the number of login attempts exceeds the *Lockout Threshold*. We observe one website, `bestbuy.com`, that has implemented such mechanisms. More sophisticated solutions based on machine learning of user login patterns are also promising.

Finally, it is crucial for all websites to enforce stronger passwords, *e.g.*, by requiring complex combinations of different types of characters, requiring minimum password length, and breaking its connection with personal information. As such, imperfection of rate-limiting mechanisms would not hinder the security of the web users.

6 LIMITATIONS AND DISCUSSION

We summarize limitations of our measurement study and discuss future improvements as follows.

Technical issues in data collection. Among the Alexa Top 500 websites, we had to exclude 71 websites from our analysis because of some technical issues. First, some websites require our cloud-based program to solve a CAPTCHA on the first visit. This situation, however, does not happen when we repeated the same experiments on machines running in our lab. It suggests some websites explicitly blacklisted the cloud IP range and disallowed user authentication requests sent from these IP addresses. Therefore, we could not

perform the analysis on these websites using the same methodology, because our method requires multiple IP address to assess the *Counting Mechanism* of the websites. Second, one of the websites has the ability to detect logins from robot, because the web server responded differently when we used Selenium scripts to log in instead of manually. Third, the rate-limiting mechanisms implemented on some other websites are different from our assumed model. For instance, on a handful of websites, we observed that after a few failed login attempts, the websites treated correct passwords as incorrect, thus we were unable to differentiate these successful and unsuccessful logins. On two other websites, we observed that login throttling was implemented by slowing down the HTTP responses without explicitly locking down the account or displaying a CAPTCHA. Our model could not handle these corner cases. These technical issues suggest our experiments can be improved by using non-cloud IP addresses and alternative web automation engines. Our study would also benefit from a more complex model with which *Counting Mechanism* can be implemented. We plan to leave them as future work.

Account lockout and login throttling. Our study did not differentiate account lockout and login throttling, as both methods thwart efficient targeted online password cracking. The intention of our research is to identify susceptible websites that allow unblocked password guessing, with the assumption that either account lockout or login throttling would mitigate automated attacks to some extent. Nevertheless, there are differences between these two mechanisms. Specifically, if a lockout mechanism is triggered, the adversary will have to wait for an extended period or manually unlock the account before further online attacks can be performed. But to deal with login throttling mechanisms, such as CAPTCHA [11, 45], the adversary can either manually solve the CAPTCHA puzzles or break it in an automated manner [46], and then proceed the attacks.

This paper can be extended in two directions. *First*, lockout mechanisms can be further investigated. The method to unlock an account can be taken into consideration when evaluating a website's susceptibility to online password attacks. For example, if the account is automatically unlocked after a short period (*e.g.*, 5 minutes), the adversary can continue the attacks after a short delay. It is also meaningful to uncover how repeated account lockout will trigger alarms to website administrators who may take further actions on such activities. However, doing so without negatively impacting the websites' regular operations is difficult. *Second*, our study can be integrated with state-of-the-art CAPTCHA cracking techniques. If the primary means for rate-limiting is by using CAPTCHAs, it is necessary to evaluate the difficulty of breaking the CAPTCHAs of these websites, and how continued logins will be affected after the CAPTCHAs have been solved. Orthogonal to our study is the protection to CAPTCHA itself. As CAPTCHA implementations in websites are diverse, various studies of CAPTCHA security [3, 8–10, 12, 16, 22, 28, 29, 36, 48] need to be combined with ours for a more comprehensive understanding of websites' countermeasures against online password guessing attacks.

Imperfect models for real-world implementations. As mentioned in Sec. 3, we have made many assumptions on the real-world implementation of authentication rate-limiting mechanisms. However, not all assumptions perfectly reflect practical implementation choices. For instance, some websites allow more than 100 failed authentication attempts and set a higher threshold instead. Our measurement study cannot identify such websites. Moreover, some websites may not strictly enforce the same threshold for a specific login interval. Some variations have been observed in our study, which have been addressed by conducting the experiments multiple times to compute the average. In summary, it is difficult to devise one model that works for all possible rate-limiting implementations. Future work would study the corner cases and design new models for those websites.

Responsible disclosure. We have tried our best to inform the susceptible websites about our study in August 2018. Among the 131 susceptible websites, only 15 websites provide an email for reporting technical/security issues; 51 require filling an online form; and the remaining 65 do not have a way for contacting them in regards to security concerns—only contacts of customer support could be found. We have contacted all 66 (15 + 51) websites that provide either email addresses or online forms. We have also notified 21 out of 28 websites that are vulnerable to Authentication Denial-of-Service Attacks, which we were able to reach. By mid September, 3 websites responded to our study[5]. For those that are less responsive, we conjecture some of them do not take rate-limiting issues as vulnerabilities (*e.g.*, `squarespace` [37]). But we hope our study could at least raise awareness of such issues to these websites.

7 CONCLUSION

In this paper, we presented a black-box approach to exploring the authentication rate-limiting mechanisms implemented by websites for preventing online password guessing attacks. Our method first modeled the rate-limiting implementation using a parameterized model and then validated the model in a sequence of black-box tests. This black-box approach can determine how a website recognizes the authentication requests from the same users and how many times a user is allowed to log in unsuccessfully before being blocked. We applied this approach to 182 websites selected from Alexa Top 500 websites and found that 131 out of the 182 websites do not properly implement the rate-limiting mechanisms: 112 of them permit repeated logins with $itvl = 1024s$; another 19 of them allow unlimited login attempts if the adversary is able to change his IP address repeatedly. Moreover, 28 of the remaining 51 websites may block a legitimate user with correct passwords when the account is locked out, effectively enabling authentication denial-of-service attacks. The study reveals susceptibility of these websites to online password guessing attacks and calls for the proper implementation of countermeasures.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their valuable comments. We also thank our shepherd, Martin Johns, for helping us improve our paper. This work is supported in part by Natural Science Foundation under Grant 1718084, 1750809, 1834215.

REFERENCES

[1] Alexa. Top Sites in United States - Alexa. https://www.alexa.com/topsites/countries/US.

[5] `Github, Canvas, Twitter.`

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

[3] Paul Baecher, Niklas Büscher, Marc Fischlin, and Benjamin Milde. 2011. Breaking reCAPTCHA: a holistic approach via shape recognition. *Future challenges in security and privacy for academia and industry* (2011).

[4] Erick Bauman, Yafeng Lu, and Zhiqiang Lin. 2015. Half a Century of Practice: Who Is Still Storing Plaintext Passwords?. In *Proceedings of the 11th International Conference on Information Security Practice and Experience*. Beijing, China.

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

[6] Joseph Bonneau and Sören Preibusch. 2010. The Password Thicket: Technical and Market Failures in Human Authentication on the Web. In *WEIS*.

[7] Sacha Brostoff and M. Angela Sasse. 2003. "Ten strikes and you're out": Increasing the number of login attempts can improve password usability. In *CHI 2003 Workshop on HCI and Security Systems*.

[8] Elie Bursztein, Jonathan Aigrain, Angelika Moscicki, and John C. Mitchell. 2014. The End is Nigh: Generic Solving of Text-based CAPTCHAs.. In *8th USENIX Workshop on Offensive Technologies*. USENIX Association.

[9] Elie Bursztein, Matthieu Martin, and John Mitchell. 2011. Text-based CAPTCHA strengths and weaknesses. In *The 18th ACM conference on Computer and communications security*. ACM.

[10] Elie Bursztein, Angelique Moscicki, Celine Fabry, Steven Bethard, John C Mitchell, and Dan Jurafsky. 2014. Easy does it: more usable CAPTCHAs. In *The 32nd annual ACM conference on Human factors in computing systems*. ACM.

[11] CAPTCHA. The Official CAPTCHA Site. http://www.captcha.net.

[12] Claudia Cruz-Perez, Oleg Starostenko, Fernando Uceda-Ponga, Vicente Alarcon-Aquino, and Leobardo Reyes-Cabrera. 2012. Breaking reCAPTCHAs with unpredictable collapse: Heuristic character segmentation and recognition. *Pattern Recognition* (2012).

[13] Anupam Das, Joseph Bonneau, Matthew Caesar, Nikita Borisov, and XiaoFeng Wang. 2014. The Tangled Web of Password Reuse. In *The Network and Distributed System Security Symposium*. Internet Society.

[14] Rachna Dhamija and Adrian Perrig. 2000. DéJà Vu: A User Study Using Images for Authentication. In *9th Conference on USENIX Security Symposium*. USENIX Association.

[15] Dinei Florêncio, Cormac Herley, and Paul C. van Oorschot. 2014. An Administrator's Guide to Internet Password Research.. In *28th Large Installation System Administration Conference*. USENIX Association.

[16] Gaurav Goswami, Brian M Powell, Mayank Vatsa, Richa Singh, and Afzel Noore. 2014. FaceDCAPTCHA: Face detection based color image CAPTCHA. *Future Generation Computer Systems* (2014).

[17] Paul A. Grassi, Elaine M. Newton, Ray A. Perlner, Andrew R. Regenscheid, James L. Fenton, William E. Burr, Justin P. Richer, Naomi B. Lefkovitz, Jamie M. Danker, Yee-Yin Choong, Kristen K. Greene, and Mary F. Theofanos. http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-63b.pdf, Title = Digital Identity Guidelines: Authentication and Lifecycle Management.

[18] Aaron L-F Han, Derek F Wong, and Lidia S Chao. 2014. Password cracking and countermeasures in computer security: A survey. *arXiv preprint arXiv:1411.7803* (2014).

[19] Briland Hitaj, Paolo Gasti, Giuseppe Ateniese, and Fernando Perez-Cruz. 2017. Passgan: A deep learning approach for password guessing. *arXiv preprint arXiv:1709.00440* (2017).

[20] Anil Jain, Lin Hong, and Sharath Pankanti. 2000. Biometric Identification. *Commun. ACM* (2000).

[21] Patrick Gage Kelley, Saranga Komanduri, Michelle L. Mazurek, Richard Shay, Timothy Vidas, Lujo Bauer, Nicolas Christin, Lorrie Faith Cranor, and Julio Lopez. 2012. Guess Again (and Again and Again): Measuring Password Strength by Simulating Password-Cracking Algorithms. In *IEEE Symposium on Security and Privacy*. IEEE Computer Society.

[22] Shujun Li, S Shah, M Khan, Syed Ali Khayam, Ahmad-Reza Sadeghi, and Roland Schmitz. 2010. Breaking e-banking CAPTCHAs. In *The 26th Annual Computer Security Applications Conference*. ACM.

[23] Yue Li, Haining Wang, and Kun Sun. 2016. A Study of Personal Information in Human-chosen Passwords and Its Security Implications. In *The IEEE Conference on Computer Communications*. IEEE.

[24] J. Ma, W. Yang, M. Luo, and N. Li. 2014. A Study of Probabilistic Password Models. In *IEEE Symposium on Security and Privacy*.

[25] Michelle L. Mazurek, Saranga Komanduri, Timothy Vidas, Lujo Bauer, Nicolas Christin, Lorrie Faith Cranor, Patrick Gage Kelley, Richard Shay, and Blase Ur. 2013. Measuring Password Guessability for an Entire University. In *ACM SIGSAC Conference on Computer and Communications Security*. ACM.

[26] William Melicher, Blase Ur, Sean M. Segreti, Saranga Komanduri, Lujo Bauer, Nicolas Christin, and Lorrie Faith Cranor. 2016. Fast, Lean, and Accurate: Modeling Password Guessability Using Neural Networks. In *25th USENIX Security Symposium*. USENIX Association.

[27] Microsoft. Microsoft Azure: Cloud Computing Platform & Services. https://azure.microsoft.com/.

[28] Manar Mohamed, Song Gao, Nitesh Saxena, and Chengcui Zhang. 2014. Dynamic cognitive game captcha usability and detection of streaming-based farming. In *The Workshop on Usable Security*.

[29] Marti Motoyama, Kirill Levchenko, Chris Kanich, Damon McCoy, Geoffrey M Voelker, and Stefan Savage. 2010. Re: CAPTCHAs-Understanding CAPTCHA-Solving Services in an Economic Context.. In *USENIX Security Symposium*.

[30] Arvind Narayanan and Vitaly Shmatikov. 2005. Fast Dictionary Attacks on Passwords Using Time-space Tradeoff. In *12th ACM Conference on Computer and Communications Security*. ACM.

[31] Benny Pinkas and Tomas Sander. 2002. Securing Passwords Against Dictionary Attacks. In *9th ACM Conference on Computer and Communications Security*.

[32] Christopher Collins Rafael Veras and Julie Thorpe. 2014. On the Semantic Patterns of Passwords and their Security Impact. In *The Network and Distributed System Security Symposium*. Internet Society.

[33] Karen Renaud, Rosanne English, Thomas Wynne, and Florian Weber. 2014. You Have Three Tries Before Lockout. Why Three?.. In *HAISA*.

[34] Selenium. Selenium - Web Browser Automation. www.seleniumhq.org/.

[35] Shadowsocks. Shadowsocks - A secure socks5 proxy. https://shadowsocks.org/.

[36] Suphannee Sivakorn, Iasonas Polakis, and Angelos D Keromytis. 2016. I am robot:(deep) learning to break semantic image captchas. In *2016 IEEE European Symposium on Security and Privacy*. IEEE.

[37] Squarespace. Reporting Vulnerabilities. https://www.squarespace.com/security.

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

[39] Paul C. Van Oorschot and Stuart Stubblebine. 2006. On Countering Online Dictionary Attacks with Login Histories and Humans-in-the-loop. *ACM Trans. Inf. Syst. Secur.* (2006).

[40] Rafael Veras, Christopher Collins, and Julie Thorpe. 2014. On Semantic Patterns of Passwords and their Security Impact.. In *NDSS*.

[41] Luis Von Ahn, Manuel Blum, Nicholas J Hopper, and John Langford. 2003. CAPTCHA: Using hard AI problems for security. In *International Conference on the Theory and Applications of Cryptographic Techniques*. Springer.

[42] Chun Wang, Steve TK Jan, Hang Hu, Douglas Bossart, and Gang Wang. 2018. The Next Domino to Fall: Empirical Analysis of User Passwords across Online Services. In *Proceedings of the Eighth ACM Conference on Data and Application Security and Privacy*. ACM.

[43] Ding Wang, Zijian Zhang, Ping Wang, Jeff Yan, and Xinyi Huang. 2016. Targeted Online Password Guessing: An Underestimated Threat. In *ACM SIGSAC Conference on Computer and Communications Security*. ACM.

[44] M. Weir, S. Aggarwal, B. d. Medeiros, and B. Glodek. 2009. Password Cracking Using Probabilistic Context-Free Grammars. In *30th IEEE Symposium on Security and Privacy*.

[45] Wikipedia. CAPTCHA-Wikipedia. https://en.wikipedia.org/wiki/CAPTCHA.

[46] Wikipedia. CAPTCHA-Wikipedia:circumvention. https://en.wikipedia.org/wiki/CAPTCHA#circumvention.

[47] Yinqian Zhang, Fabian Monrose, and Michael K Reiter. 2010. The security of modern password expiration: An algorithmic framework and empirical analysis. In *The 17th ACM conference on Computer and communications security*. ACM.

[48] Bin B Zhu, Jeff Yan, Qiujie Li, Chao Yang, Jia Liu, Ning Xu, Meng Yi, and Kaiwei Cai. 2010. Attacks and design of image recognition CAPTCHAs. In *The 17th ACM conference on Computer and communications security*. ACM.

[49] Chaoshun Zuo, Wubing Wang, Rui Wang, and Zhiqiang Lin. 2016. Automatic Forgery of Cryptographically Consistent Messages to Identify Security Vulnerabilities in Mobile Services. In *Proceedings of the 23rd Annual Network and Distributed System Security Symposium (NDSS'16)*. San Diego, CA.