Araña: Discovering and Characterizing Password Guessing Attacks in Practice

Mazharul Islam*†, Marina Sanusi Bohuk*‡, Paul Chung†, Thomas Ristenpart‡, Rahul Chatterjee†
† *University of Wisconsin–Madison* ‡ *Cornell Tech*

Abstract

Remote password guessing attacks remain one of the largest sources of account compromise. Understanding and characterizing attacker strategies is critical to improving security, but doing so has been challenging thus far due to the sensitivity of login services and the lack of ground truth labels for benign and malicious login requests. We perform an in-depth measurement study of guessing attacks targeting two large universities. Using a rich dataset of more than 34 million login requests to the two universities as well as thousands of compromise reports, we were able to develop a new analysis pipeline to identify 29 attack clusters—many of which involved compromises not previously known to security engineers. Our analysis provides the richest investigation to date of password guessing attacks as seen from login services. We believe our tooling will be useful in future efforts to develop real-time detection of attack campaigns, and our characterization of attack campaigns can help more broadly guide mitigation design.

1 Introduction

Remote password guessing attacks are one of the most effective and prevalent causes of account compromise for password-based authentication systems [43, 48]. Password guessing attacks are easy to mount for attackers, who may attempt logging in under known usernames with widely popular passwords or perform credential stuffing by submitting username-password pairs that have appeared in previous breaches. Despite being straightforward, such attacks can nevertheless be highly damaging. As a result, the most advanced login services in practice use proprietary mechanisms in an attempt to detect malicious logins using more than just the correctness of the submitted password, e.g., via user risk profiles [18, 19, 43].

Improving such mechanisms requires understanding attacker behavior as observed by login services—a delicate task given the sensitive nature of login data, especially passwords. As such, no in-depth measurement studies of attack behavior as seen from login services have been conducted.

Recently, in our prior work [7], we designed a new login service instrumentation tool called Gossamer. It securely records information about login requests, including certain carefully chosen statistics about the passwords used in the requests. Gossamer was deployed for seven months at one university (U1) and three months at another (U2). We analyzed various aspects of user login behavior, but were only able to identify three obvious, high-volume attacks. How to find and characterize more attacks was left as a tantalizing open question.

In this work, we make progress on answering this open question. To do so, we obtained approval—from our university IRBs and IT security teams—to perform a fresh analysis of the datasets generated by the previous study. These datasets include a rich amount of information on over 34 million login requests, Duo two-factor authentication (2FA) logs (just for U2), and more than 2,000 compromise reports.

Even with the measurements provided by Gossamer, discovering attacks represents a tricky bootstrapping problem. The compromise logs do not implicate specific login requests and, more broadly, there is no ground truth anywhere in the dataset for attacks. This makes the supervised machine learning approaches used in prior work [18, 19] inapplicable to our setting. At the same time, the scale of the problem renders manual analysis prohibitive, and so prior work only used a simple, known method for detecting attacks: Flag IP addresses that individually flooded the login service with requests. This of course misses many attacks.

We develop a new analysis pipeline that we call Araña. Crucially, it focuses not on individual requests but sets of login requests emanating from the same IP within a day. This ensures sufficient signal about client behavior for patterns to emerge. We then built an application-specific filtering, clustering, and manual analysis approach. We filter out likely-benign login request sets using custom heuristics, such as filtering out IP addresses exhibiting high login success rates. We then

*co-first authors

use an unsupervised learning approach, specifically agglomerative clustering [33], with a custom distance function tailored to login sets. This helps us identify clusters of login sets that can be manually analyzed with little effort to discover *attack campaigns*—sets of login requests likely to be submitted by the same attacker.

Araña is effective at finding these attack campaigns, including coordinated attacks involving multiple IP addresses and those spread out over long stretches of time. We use it to discover and characterize a diverse set of 29 attack clusters. Many are high volume credential stuffing attacks that submit on average one password per targeted username and exhibit a notable fraction of username-password pairs appearing in breaches. In addition to the more simplistic stuffing attacks that quickly flood requests from a handful of IP addresses, Araña allowed for discovering widely distributed attacks that may originate from hundreds of different IP addresses. Some of these attacks try to be "low and slow", submitting a low number of requests per day at a slow rate. This confirms that credential stuffing attackers attempt to evade countermeasures that focus on individual high volume IPs. These credential stuffing attacks are unfortunately effective: We uncover hundreds of potential successful logins by attackers for usernames not flagged already in compromise reports prepared by security engineers.

We are also able to discover lower volume, targeted attacks by focusing on Araña-identified clusters that exhibit a large number of requests to individual usernames. For example, we identified an attack campaign made up of two clusters that targeted 127 users with on average 25 guessed passwords per user. These attacks included successful logins, suggesting this targeted strategy can also work for attackers.

We discuss many other attack campaigns in the body. Overall, our analyses highlight a number of important takeaways for authentication system designers. First, they suggest that, perhaps unsurprisingly, credential stuffing attacks remain a primary vector for account compromise. This, plus the fact that most compromised users in our study also had passwords in known breaches, underscores the urgent need for broader use of breach alerting APIs (e.g., [22, 27, 43, 45]).

Second, we saw a large number of attacks on Microsoft's basic authentication endpoint at U2. It is the least protected of all of U2's authentication services, as it does not support rate limiting or Duo 2FA. It is also easy for attackers to target any organization using basic authentication by just changing the destination URL. We observed that attackers regularly target such weak points, underlining a challenge for large organizations with heterogenous authentication infrastructure.

Third, mechanisms that look for a large number of submissions per unit time from an IP or to a username are ineffective against many attacks already being deployed in practice. A better approach would be to work towards operationalizing mechanisms like Araña to detect in (near) real time distributed and sneaky attacks. However, future work will be needed to explore how to make such mechanisms robust against evasion by future adversaries.

Summary. Our contributions include the following:

- We perform the first in-depth analysis of a dataset including over 34 million login events generated by a recent system called Gossamer, in order to discover and characterize remote password guessing attacks.
- We design an analysis framework, called Araña, that shows how to filter and cluster Gossamer logs to enable easy manual analysis and identify attack campaigns.
- We use Araña to discover and characterize 29 attack clusters against two major universities that compromised hundreds of user accounts in total.
- We identify key characteristics and patterns of attacks received by authentication systems at these universities, and we discuss how authentication systems should evolve to counter such threats.

Our work has already had some practical impact in terms of discovering new attacks. We have worked with security engineers from the two universities to perform responsible disclosure of potentially compromised usernames. We believe that Araña will be useful in developing more robust attack detection methods, and we release it as a public, open source project [1].

2 Background and Related Work

Passwords remain the most widely used mechanism for user authentication, despite efforts to move past them [8]. We focus our discussion on the literature most closely related to our topic: characterizing password guessing attacks as seen by login services.

Measurement studies on user passwords. User behavior with respect to password choice has been extensively researched. Many studies simulate login services (e.g., via Mechanical Turk) to perform user studies [24, 26]. Others look at password breach data to characterize aspects of user password selection [14, 47]. A handful of studies have measured user passwords in real deployments [8, 12, 17, 30].

Most recently, the Gossamer system [7] was used to measure not only legitimate user password strength, but also various user password submission behaviors as observed by the single-sign on (SSO) login services at two large universities. This paper uses the same measurement datasets as reported in [7], but focuses on characterizing malicious login behavior.

Password guessing attacks. The literature discussed so far concerned itself with legitimate user behavior, trying in part to assess whether users are selecting passwords that resist various types of guessing attacks. A traditional focus has been on *offline password cracking attacks*, which occur when an attacker attempts to crack password hashes found in exposed password hash databases using tools like Hashcat [40] and John the Ripper [35]. Researchers have also developed

natural language processing techniques to improve guess generation [20, 25, 28, 32, 34, 47].

Breached username-password pairs—either obtained via offline cracking, gathered via phishing or malware, or stolen from another web services storing plaintext password—can be used in *credential stuffing attacks*, where an attacker tries to log into a system using breached username-password pairs. Users frequently reuse passwords across multiple services [14, 36, 46], making credential stuffing one of the most prevalent forms of account compromise attacks [43, 48]. In an effort to reduce credential stuffing attacks, breach alerting services such as HaveIBeenPwned [22], Google Password Checkup [44], and Cloudflare's exposed credential checks [37, 45] provide APIs that check whether a user's passwords have been compromised.

Credential stuffing is one kind of *online guessing attack*—an attack that remotely attempts to log into a service with guessed credentials. Other examples of guessing attacks include *password spraying attacks* that submit a few very popular passwords against a large number of user accounts and *credential tweaking attacks* [36] that submit slight variants of breached passwords.

Detecting malicious logins. Resisting online password guessing attacks requires determining which login attempts are malicious. As Bonneau et al. [9] discuss, login services increasingly should treat login as more of a classification problem, taking into account more than just the correctness of a submitted password. But only a handful of prior works have focused on how to do so.

Freeman et al. [18] were the first to report on a statistical framework using the client IP address and user agent to differentiate between valid and invalid login attempts; this study used real-world LinkedIn login data, but did not include password-based features. They also do not report on observed attack campaigns. Schechter et al. [39] build a malicious login detection system that also utilizes password-based features, including differentiating password typos from other failures and using privacy-preserving data structures such as a binomial ladder filter for detecting frequently used passwords. Their study used simulated data to argue the system's efficacy at detecting malicious logins. Finally, Thomas et al. [43] studied underground forums and tools used to steal credentials—their only measurements using logins are used to report on the lack of evidence of targeted guessing attacks that try multiple queries against accounts.

The Gossamer paper [7] reports on three attack campaigns that are easily detected using known techniques (i.e., a huge number of requests from an IP in a relatively short amount of time). Such techniques will only catch obvious attacks and are blind to "low and slow" attacks that purposefully use a small number of guesses per target account (low) and per unit time (slow), or distributed attacks that perform guesses from many different IP addresses. While missing these attacks does not appear to affect the results on benign user behavior reported in [7], understanding attacker behavior requires finding and investigating these attacks.

In summary, no prior work has characterized the behavior of password guessing attacks as seen from login services.

Two-factor authentication. Although effective at preventing account compromise, two-factor authentication (2FA) has yet to achieve widespread adoption [13, 15], and user friction is still very high [4, 16]. As we show, many older accounts at universities are not enrolled in 2FA; and more importantly, even when 2FA is used, recent attacks have shown that attackers can spoof push-based 2FA and hide spoofed pushes by sending them soon after the victim has logged in [23]. Regardless of whether 2FA is bypassed in an attack, we would like to know when an attacker successfully guesses a user's password so that IT security personnel and/or the user can take preventative steps such as changing the password for the indicated account and for any other account that uses the same (or a similar) password.

Unsupervised techniques for attack detection. In this work, we therefore develop a new approach for detecting and characterizing password guessing attack campaigns using Gossamer logs. As we explain in Section 4.2, we did not have success using supervised machine learning approaches and so instead focus on unsupervised techniques, which have a long history of use in attack detection. For example, they are frequently used in intrusion detection systems (c.f., [10]) and for various kinds of anomaly detection (c.f., [5]). To the best of our knowledge, no prior work has developed mechanisms to detect whether password-based logins are malicious.

3 Gossamer Logs

In this work, we use a dataset of login requests compiled from two universities (U1 and U2) via our prior work Gossamer [7]. For completeness, we provide some details about Gossamer and how the resulting datasets were collected. We refer readers to [7] for more details.

Gossamer logs. Our prior work introduced a new, privacy-preserving instrumentation approach for use with login systems [7]. The resulting system, Gossamer, provides a secure way to collect measurement statistics about login requests, including a subset of HTTP headers, source IP address, target username, success or failure of a login request, and carefully chosen measurements on the submitted password. Gossamer uses two levels of storage to provide extra security for particularly sensitive information. The submitted passwords are encrypted and stored in an in-memory database. They are cryptographically erased every 24 hours, providing a good trade-off between the ability to calculate password statistics over one-day windows (e.g., the edit distance between two submitted passwords) and the ability to protect the secrecy of passwords even in the low-likelihood case of a full compromise of the instrumentation service. The system also deterministically encrypts the usernames to preserve their privacy

and blind researchers from them.

Gossamer was designed in close collaboration with the information technology (IT) security teams at two large universities. We received approval from the IRB and IT security teams to deploy Gossamer for 7 months at U1 and for 3 months at U2. In total, we collected more than 34 million login requests from 347 K valid usernames. In [7], Bohuk et al. detailed the design of Gossamer and reported on an analysis of this dataset in terms of characterizing legitimate user behavior. At the time we were only able to detect a few obvious attacks that stood out due to their high rate of requests, and left finding more attacks as an open question. This paper is a first step at addressing this open question.

Duo logs. Both universities use Duo 2FA for most login requests; users are prompted to provide this second factor after they successfully submit the password (and if they do not have a valid "Duo cookie" stored in their browser). The datasets we received include sanitized Duo logs for U2 only.

Unfortunately, there is no identifier in the Duo logs that can be used to uniquely associate a log entry with a particular login request. We therefore had to use a timestamp-based heuristic similar to that used previously in [7], to find the Duo prompt that likely corresponds to a successful password submission. Using the timestamp and encrypted username associated with the Duo request, we correlate the Duo requests with login requests within a 2-minute time window. Out of these requests, 96.7% were successfully completed, 3.2% were denied, and only 46 ($< 0.001\%$) were marked as fraud by the user.

Compromised user logs. At both universities, the security analysts have processes for identifying and reporting compromised accounts. These processes include user reporting mechanisms and third-party breach notification services (e.g., U1 uses a Microsoft product to detect accounts that are sending spam). We were given access to compromised account logs for the period of our previous measurement study, containing the encrypted username, the timestamp reported, the estimated timestamp of the attack (only available at U1), and the reason for compromise. Importantly, compromise reports at U1 were logged from multiple authentication services, but the Gossamer deployment only instrumented the main one (U1 web login); so the number of compromised accounts will be an upper bound for those that were actually compromised through U1 web login. Also, the IT departments consider an account compromised if the attacker made a successful login request, even if they did not bypass two-factor authentication. Indeed, many older accounts may not be enrolled in 2FA. We adopt the same terminology and refer to account compromises as a successful login from an attacker independent of whether 2FA was also bypassed. In most cases, analysts respond to compromise reports by scrambling the passwords of compromised accounts to prevent further damage.

Our prior work using Gossamer logs contains some basic details about the compromise logs; we elaborate more here since it will be relevant to interpreting some results later in the paper. We considered compromised reports for the measurement time period at each university, plus one additional week (which allowed time for analysts to enter compromise reports for attacks that may have occurred during the measurement period). We give summary statistics on this compromise database in Figure 1. On average, 190 usernames (0.59% of all valid usernames) are reported every month at U1, and 163 usernames (0.28% of all valid usernames) at U2. At U1, 323 usernames were reported multiple times, 32 of which were reported for two different reasons. Users who are reported twice are compromised on average 26 days apart.

	U1 (7 mo.)	U2 (3 mo.)
Total reports	1,818	611
Unique usernames		
- reported	1,468	489
- more than once for same reason	323	122
- more than once for diff. reasons	32	0
% of unique usernames compromised / month	0.59%	0.28%
# IPs tried to log in		
- with a compromised username	4,633	6,409
- with multiple compromised usernames	716	156
Max. comp. usernames associated with an IP	382	24
Avg. usernames associated with a single IP	1.98	1.05

Figure 1: Summary statistics on the compromised accounts reported at each university during the measurement period.

We discuss a breakdown of the reasons for compromise in more detail in Appendix A. In summary, we found that the majority of accounts were reported as compromised through large-scale automated attacks. At U1, the estimated time of attack is also reported for each compromised user, and so we investigated the time it takes for an attack to be recorded in the compromise database. We found that 17% of compromised accounts are reported within the first hour after the attack, and 90% of compromised accounts are reported within 61 hours. However, the "timestamp of attack" field is an approximation of the time of attack based on the analyst's best guess. There is no such field recorded at U2.

We also approximate the time of attack by taking the last successful login for a given username before that username is entered into the account compromise database. In doing so, we find that only 4% of compromised accounts at U1 are reported within the first hour after their last successful login, and 90% are reported within 46 days. This large difference shows that either the analyst's estimated time of attack was not very accurate, or the last successful login before a compromise report is not a good approximation for the time of an attack—an important challenge in using compromised logs for detecting attacks, as we discuss in the next section.

Ethical considerations. Throughout our analysis, we worked closely with the IT security offices at both univer-

sities similar to how we did for Gossamer [7]. We obtained approval for both this study and Gossamer from the IT security offices and university administrators. Both studies were reviewed by our university IRBs and received IRB exemptions, as the Gossamer logs do not include usernames or other data considered to be PII by IRB. We could not request consent from individual users, as we do not know the usernames of users. To further protect the privacy of users, we used similar security measures as we used for Gossamer study for data analysis: We used a machine only accessible to a subset of researchers logging in from a specific computer in the university network and requiring a strong password and 2FA for access. We also notified the IT administrations at both universities of all previously unreported compromised accounts we found using the encrypted usernames.

4 Towards Detecting Attack Campaigns

We now turn to the challenge of discovering remote password guessing attacks using Gossamer logs. Prior work has used supervised machine learning approaches to flag likely malicious login requests [18,39]. However, they relied on simulated logins, for which they marked each request as benign or malicious.

For login requests observed in practice, flagging each as either benign or malicious represents a challenging bootstrapping problem. Not only is the number of login requests received too massive for a reasonable set of security engineers to manually analyze, but also there is no clear set of criteria to flag a login request as malicious. Moreover, even if an account has been flagged as compromised by a security engineer or reported as compromised by a user, there is no obvious way to correlate this with individual login attempts because, as discussed in Section 3, compromise databases do not include information on the specific login sessions.

Individual login requests often do not contain enough information for even a human analyst to determine if they are malicious. We instead aim to identify *attack campaigns*—that is, sets of login requests submitted by the same attacker. Attackers often use automated scripts to send guesses for usernames and passwords from one or more IP addresses over a period of hours, days, or weeks. To identify attack campaigns we will cluster login requests into groups that are likely to be part of the same attack campaign, as we explain below.

4.1 Login Sets and Features

The main Gossamer logs consist of a sequence of login requests. Each request entry includes (1) a timestamp; (2) the (deterministically encrypted) username; (3) the client IP address; (4) the client user agent string; (5) whether the submitted password is weak (has a bucketized zxcvbn score of zero, as explained in prior work [7]); (5) the edit distances between the submitted password and previous passwords submitted by the same IP or username; (6) whether the request succeeded or failed due to an invalid username or the wrong password; (7) whether the submitted password is a tweaked variant of a breached password known to Gossamer; or (8) whether the submitted password, username, or username-password pair appear in a breach dataset known to Gossamer.

Group	Features	Acronym
Client	IP address ♦	IP
	ISP ♦	ISP
	User agent string ♦	UA
Volume	Total # requests submitted	NR
	Total # unique usernames submitted	NU
	Avg. # unique password per user	AUP
Login timing	Date♦	DATE
	Mean interarrival time	MIT
	Std. interarrival time	SIT
Success rates	Fraction of submitted requests	
	that failed	FF
	w/ invalid usernames	FIU
Password guessability	Fraction of submitted requests	
	w/ a weak pw, zxcvbn$(w) = 0$	FWP
	w/ pw in breach	FPIB
	w/ username in breach	FUIB
	w/ username-password pair in breach	FCIB
	w/ a tweaked pw in breach	FTP

Figure 2: Features for $\mathcal{L}$ sets that we use for analysis. Nominal features are marked with ♦; all others are numerical.

We group login requests based on the client IP address and date it was received. We call this grouping a *login set*, which we denote by $\mathcal{L}$. The set of all $\mathcal{L}$ sets within a Gossamer log we denote as $\mathbb{L} = \{\mathcal{L}_1, \ldots, \mathcal{L}_n\}$, where n is the number of $\mathcal{L}$ sets in the log. Each $\mathcal{L}_i$ contains all login requests received in a day from an IP address. Note that $\mathbb{L}$ defines a partition over all login attempts.

We define an *attack campaign* as a set of one or more $\mathcal{L}$ sets. This assumes that all logins in a $\mathcal{L}$ set are either malicious or all are benign. Although it is possible that legitimate users might share the same VPN or proxy IP address as an attacker, we expect such scenarios to be rare (and did not encounter them in our analyses). We leave to future work how to differentiate benign and attack login attempts from a single IP.

To determine if an $\mathcal{L}$ set is potentially malicious or benign, we utilize a rich set of features describing a $\mathcal{L}$ set based on Gossamer logs. The full set of features, consisting of four nominal and 12 numerical features, are summarized in Figure 2. At a high level, these 16 features can be roughly divided into following groups:

- *Client features* include the source IP address and user agent (UA) within the request. We also look up the ISP associated with each IP address (as reported via the MaxMind Geolocation API [29]). These are useful for determining whether requests are from the same client device.
- *Volumetric features* include the total number of requests in a $\mathcal{L}$ set (NR), the number of unique usernames targeted in

the set (NU), and the average number of unique passwords submitted to a particular username (AUP) for a $\mathcal{L}$ set.

- *Login timing features* include the date of the $\mathcal{L}$ set, as well as the mean (MIT) and standard deviation (SIT) of the interarrival time between requests within the set.
- *Success rate features* measure the fraction of invalid usernames submitted (FIU) and the fraction of invalid username-password pairs submitted (FF).
- *Password guessability features* include the fraction of submitted passwords in an $\mathcal{L}$ set that have a zxcvbn score of zero (FWP), indicating a weak password, as well as the fraction of submitted passwords in a known breach (FPIB), usernames in a known breach (FUIB), username-password pairs in a known breach (FCIB), and the fraction of passwords submitted that are a close variant of a breach password—called a "tweaked password" (FTP).

Together, these features serve as the basis for our attack campaign detection mechanisms.

4.2 Initial Attempts Using Compromise Reports

An obvious potential approach for detecting attack campaigns is to utilize compromised account reports to label $\mathcal{L}$ sets as potentially malicious. However, as discussed in Section 3, those reports do not record information sufficient to identify exactly which $\mathcal{L}$ set contains the compromising login attempt. We note that this ambiguity is not just a limitation at the universities we investigated; anytime a login system allows compromise reports from users or third party services, they will not be directly associated with logins. Nevertheless, we experimented with various ways of using compromise reports as ground truth for supervised approaches. We briefly report on two here.

Classifier based approaches. We identified 23,016 $\mathcal{L}$ sets where the corresponding IP address contacted at least one eventually-compromised user account (at any point during the data collection period). Let $\mathcal{C}$ denote the set of all those $\mathcal{L}$ sets. Of course, not all of these are necessarily malicious; but we mark them all as malicious since it is unclear how to label them more precisely.

We first tried to develop a classifier that can identify $\mathcal{L}$ sets likely to have a compromised user based on the features we identified in Figure 2. We added to $\mathcal{C}$ an equal number of $\mathcal{L}$ sets not associated with a compromised user account, labeled them as benign, and performed an 80/20 training and testing split on the combined set. We then trained linear regression, decision tree, random forest, logistic regression, and SVM models to predict maliciousness. All the models exhibited very bad precision; the linear regression classifier performed best, with a recall 0.79 but a precision of just 0.13, making it essentially useless. The primary reason is the crude labeling of training data as malicious or benign, presumably including many false positives.

Directed anomaly scoring. We also explored using Ho et al.'s directed anomaly scoring (DAS) [21], tuned via compromise reports, to rank $\mathcal{L}$ sets in order of "suspiciousness." These experiments were promising at detecting new types of attacks, but failed to help us detect attacks spanning multiple source IP addresses, making it less useful for driving an analysis and characterization of attack campaigns. See Appendix B for details.

5 Campaign Discovery Pipeline

We developed an analysis pipeline, called Araña, that combines heuristic-based filtering, unsupervised machine learning to cluster behavior, and a manual analysis review step as illustrated in Figure 3. We refer to this as an FCA (filter, cluster, analyze) approach. While the general notion of an FCA-type analysis pipeline is not novel, our application-specific heuristics, similarity measures, and manual review processes are new. Our goal here is not completeness; identifying all malicious logins is impossible. Rather, we build a high precision (low false positive) pipeline that helps us discover a wide range of attack campaigns with minimal manual effort.

We first work to develop heuristics for filtering out likely benign $\mathcal{L}$ sets. First, we remove $\mathcal{L}$ sets that do not exhibit a high failure rate (HFR), and then we remove some anomalous known-benign behaviors (such as misconfigured clients repeatedly making failed requests).

We then use the wide range of features described in the previous section to help us define an application-specific similarity measure $\mathsf{logsetsim}(\mathcal{L}_i, \mathcal{L}_j)$ that outputs a similarity score for any $\mathcal{L}$ set pair $\mathcal{L}_i, \mathcal{L}_j$. This helps us capture the likelihood that two $\mathcal{L}$ sets are part of the same attack campaign. Given logsetsim, we can create a pairwise distance matrix and apply unsupervised agglomerative clustering [33] to discover clusters of $\mathcal{L}$ sets that could each be part of a single attack campaign. The candidates can then be manually inspected by analysts; we report on our findings in Section 6.

5.1 Filtering Likely Benign Requests

Most logins are benign, and benign requests are a source of obfuscating noise in unsupervised algorithms. We therefore first develop a set of heuristics for filtering out likely benign behavior or, equivalently, identifying potentially malicious $\mathcal{L}$ sets. Our heuristics are based on three assumptions regarding attacker behavior reported in prior work [18, 19, 39]: (a) malicious logins are only a small fraction of the total login traffic; (b) a large fraction of the malicious login attempts fail; and (c) attackers use automated scripts or tools to minimize the time and effort required to send a large number of requests from IP pools (purchased or free proxies, VPNs), hiding their own IP. Based on these assumptions, we first use heuristics to filter out obviously benign behaviors (thereby identifying potentially malicious behavior).

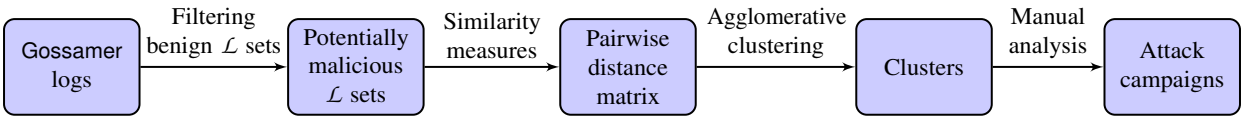

Figure 3: Araña's filter, cluster, analyze pipeline for discovering attack campaigns.

Filtering based on failure rate. We start by focusing attention on $\mathcal{L}$ sets that exhibit a high failure rate (HFR). We choose thresholds for two features: the total number of login requests NR within $\mathcal{L}$ and the fraction FF of these login requests that are failures. We say a $\mathcal{L}$ set meets the HFR condition if NR $> l$ and FF $> f$ for two thresholds l and f that we set below. Intuitively, benign logins should succeed most of the time, and failure rates are not meaningful for one or two logins.

To choose these thresholds l and f, we first plot the distributions of NR and FF over the subset of $\mathbb{L}$ that includes all $\mathcal{L}$ sets that have at least two login requests (NR > 1) and one failed login request (FF > 0). We show their distributions in Figure 4. If we choose extreme thresholds, the HFR heuristic may miss potentially malicious $\mathcal{L}$ sets; but on the other hand, choosing a relaxed threshold may flag benign $\mathcal{L}$ sets as potentially malicious, increasing the noise in our final attack campaign detection procedure.

Our goal was to build a semi-automated tool for discovering high-volume attack campaigns and help IT analysts discover them with minimal manual effort. Since security analysts are concerned with finding the most damaging attacks (that is, attacks compromising the highest number of users) with high precision, we chose aggressive thresholds for considering malicious login sets. Thus, at U1, we used the 90^{th} percentile as the threshold, making $l_{U1} = 9$ and $f_{U1} = 0.77$. However, for U2, the 90^{th} percentile yielded $f_{U2} = 1.0$, the highest possible value; this is because U2 had more high volume unsuccessful attacks which inflated the 90^{th} percentile. Therefore, we decreased the threshold to the 80^{th} percentile for U2, setting $l_{U2} = 6$ and $f_{U2} = 0.8$. Although these aggressive thresholds may miss certain attacks, lowering them runs the risk of degrading the quality (precision) of clusters by mixing $\mathcal{L}$ sets with starkly different behavior in the same cluster. Lower thresholds may also cause benign clusters to be mistakenly flagged as malicious, which would quickly cause analysts to lose trust in the tool. We investigate the effects of lowering thresholds by running additional experiments with more relaxed thresholds as explained in Appendix C .

Using the HFR heuristic, we filtered out a large number of outright benign $\mathcal{L}$ sets for both universities, as shown in Figure 5. This allowed us to focus on $\mathcal{L}$ sets exhibiting more anomalous behavior and potentially malicious behavior.

Filtering out benign behavior. After removing $\mathcal{L}$ sets that do not match the HFR heuristic, we further filter out other likely benign behaviors.

First, we filter out $\mathcal{L}$ sets with IPs that have successfully

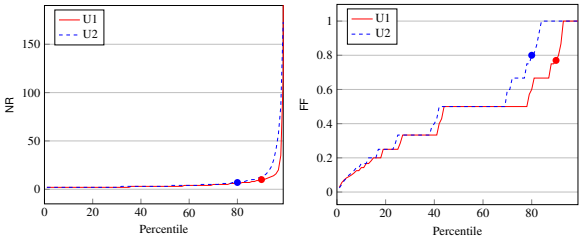

Figure 4: Percentile of NR (left) and FF (right) for both universities (shown up to the 99^{th} percentile for viewing). By choosing the 90^{th} percentile at U1 and 80^{th} at U2, we set $l_{U1} = 10$, $l_{U2} = 7$ and $f_{U1} = 0.77$, $f_{U2} = 0.8$ at the universities respectively.

completed Duo requests for all target users at least once on the same day because we assume that a remote attacker conducting large scale password guessing attacks does not have physical access to a victim's Duo authentication device. As we only have the Duo logs at U2, we applied this filtering to the U2 data. We found 563 $\mathcal{L}$ sets matching the HFR heuristic, but all of them contacted only one user and had successfully completed at least one Duo request for that user. Around 90% of these $\mathcal{L}$ sets had submitted ≤ 5 unique passwords, and the failures were due to incorrect password entry—probably just a typo.

Second, we filter out any $\mathcal{L}$ sets having an IP within the university networks or belonging to school proxy/VPN servers. Access to these IPs is restricted, and members of the two universities can only use them after successful authentication and completing the Duo request.

Lastly, we filter out $\mathcal{L}$ sets that seem to emanate from a malfunctioning or misconfigured (benign) client. Specifically, we noticed many IPs submitting a large number of failed login requests with the same incorrect password. Upon further inspection, we observe that these were automated requests belonging to email clients, Outlook Exchange Web Services, and calendar auto-sync agents that were configured with an invalid password. Since any rational attacker would not try the same invalid username-password pair repeatedly, we removed all $\mathcal{L}$ sets reusing the same incorrect username-password pair for more than 90% of their login requests.

After filtering benign $\mathcal{L}$ sets based on the HFR condition, failure rate, successful Duo two-factor submission, school IP addresses, and number of unique username-password pairs, 1,717 potentially malicious $\mathcal{L}$ sets remained at U1, and 6,408 $\mathcal{L}$ sets remained at U2. We show a breakdown of these filtering steps in Figure 5.

Filter	# $\mathcal{L}$ sets	
	U1	U2
$\mathcal{L}$ with NR > 1 & FF > 0	225,468	227,893
$\mathcal{L}$ with NR $> l$ & FF $> f$ (HFR)	2,074	11,929
$\mathcal{L}$ with HFR and		
w/ successful Duo request on DATE	n/a	563
IP belongs to school IP pool protected by Duo	60	825
>90% password reuse	419	4,133
Remaining $\mathcal{L}$ sets	1,717	6,408

Figure 5: Number of $\mathcal{L}$ sets matching each of the filtering criteria at either university. The last row shows the remaining number of sets after all filtering steps. Threshold values l and f for U1 and U2 are indicated in Figure 4.

5.2 Clustering Potentially Malicious $\mathcal{L}$ sets

After the filtering step, we cluster the remaining $\mathcal{L}$ sets using the features described in Figure 2. The key question for clustering is how to define a distance function that gauges similarity between $\mathcal{L}$ sets, such that two $\mathcal{L}$ sets have a small distance if they belong to the same attack campaign.

Similarity modeling. We design a distance function that can measure the likeliness that two given $\mathcal{L}$ sets belong to the same campaign. For the numerical features presented in Figure 2, we use the normalized difference between two values x and y, defined as $\mathsf{ND}(x,y) = |x-y|/(x+y)$. We chose this particular distance function since $\mathsf{ND}(x,y) \approx 0$ when $x \approx y$, and it is a high value when x and y have a high difference. That said, we believe other numerical distance measures would work well too.

For the nominal features IP, ISP, UA, and DATE, we cannot do a straightforward equality checking to measure their similarity, since these nominal features are sparse. Therefore, similar to Freeman et. al. [18], we use a hierarchical backoff distance (HBD) approach. This technique defines a number of levels: If the two feature values do not match at a lower-level, we "backoff" and use values of the feature from a higher level, while incurring a dissimilarity cost. We set this dissimilarity to $e^{-\ell} - e^{-(\ell+1)}$ for backing off from the lower hierarchical level ℓ to a higher hierarchical level $\ell+1$, and we accumulate the dissimilarity costs from each level with each backoff.

For the IP feature, we use the hierarchical structure of Internet routing to define four levels. At level $\ell = 0$ we check for strict IP equality; at level $\ell = 1$ we check for /24 subnet equality; at level $\ell = 2$ we check for ISP equality (as reported in the ISP feature); and at level $\ell = 3$ everything is considered equal. Thus the max level for this feature is $\ell^{\mathsf{IP}}_{\mathsf{max}} = 3$. For the UA feature, we use five levels. At level $\ell = 0$ we check if the UA strings are identical. For subsequent levels we extract from the UA string to obtain the application (desktop, mobile, unknown), browser (chrome, edge, other), and OS (Windows, iOS, Mac OS, Linux, other). Equality checks for each of these three define levels $\ell = 1$ through $\ell = 3$. The final level $\ell = 4$ indicates that the user agents do not have anything in common.

For the final nominal feature, DATE, we find the number of days d between two $\mathcal{L}$ sets and compute $1 - e^{-d}$. Thus two $\mathcal{L}$ sets with the same DATE have a distance of 0; with 1 day apart, the distance is 0.63, and so on. Note that this backoff can be calculated (less efficiently) via the same approach as for the other nominal features, by setting $\ell^{\mathsf{DATE}}_{\mathsf{max}}$ equal to the maximum number of days.

Pseudocode for the complete logsetsim and the hierarchical backoff distance (HBD) is shown in Figure 6. The set Γ includes all feature we used (shown on Figure 2), and we abuse notation slightly by letting $\gamma \in \Gamma$ define a function that maps a $\mathcal{L}$ set to the relevant feature value. For the nominal features, we further let γ_ℓ denote the function that extracts the ℓ^{th} level from the feature value. We also define a predicate type over features that indicates whether the feature is nominal or numerical. We weight the distance values computed for each feature based on a hyper-parameter called feature weight $w^\gamma \in [0,1]$.

```
logsetsim(L1, L2):
s ← 0
for γ ∈ Γ:
    x ← γ(L1); y ← γ(L2)
    if type(γ) is numerical:
        s ← s + w^γ · ND(x, y)
    else
        s ← s + w^γ · HBD(γ, x, y)
return s
```

```
HBD(γ, x, y):
s ← 0
ℓ ← 0
while ℓ < ℓ^γ_max or γ_ℓ(x) ≠ γ_ℓ(y) do
    s ← s + e^{-ℓ} − e^{-(ℓ+1)}
    ℓ ← ℓ + 1
return s
```

Figure 6: Our distance function (logsetsim) to measure the similarity between two $\mathcal{L}$ sets (Left) and the hierarchical backoff distance (HBD) calculation for nominal features (Right).

Clustering. We used an agglomerative clustering technique [33] that can work with a non-metric distance function. Agglomerative clustering starts with each data point as a single cluster and only merges two clusters if their distance is smaller than a given threshold. Adjusting the threshold can help avoid clustering together $\mathcal{L}$ sets showing different login behavior. To set the distance threshold appropriately, we rely on a knee locator method [38] frequently used in clustering algorithms to find the correct threshold for distances. We set the linkage type to "average" and set the distance threshold for U1 and U2 to 0.44 and 0.51 respectively after applying the knee locator method at each university separately. The silhouette score [6] for agglomerative clustering was 0.19 for U1 and 0.17 for U2, which beat other approaches we tried (see Appendix C).

Implementation details. We implemented our similarity model logsetsim in 240 lines of code written in Python 3.6. To extract the four hierarchical levels for the IP feature, we used the MaxMind GeoIP database [29]; and for the UA feature, we used the ua-parser package [3]. Given an $\mathbb{L}$ of size n, we computed an $n \times n$ distance matrix to be used for various

clustering algorithms. Computing the distance matrix takes $O(n^2)$ time; however, it can be easily parallelized. We used 40 threads and were able to compute the distance matrix for $n = 1,717$ within 9 minutes at U1 and for $n = 6,408$ within 28 minutes at U2 (using an Intel Xeon Linux machine with 56 cores and 125 GB of memory). For the clustering step, we used the sklearn [11] library, and clustering completed in less than a minute for both universities.

5.3 Attack Campaigns Discovered

Based on our designed similarity modeling, the agglomerative clustering approach described in Section 5.2 produced 366 clusters from 1,717 $\mathcal{L}$ sets at U1 and 640 clusters from 6,408 $\mathcal{L}$ sets at U2. For each cluster we recompute the feature values mentioned in Figure 2, after taking the union of $\mathcal{L}$ sets. Next, we sample a few of the top most interesting clusters for manually analyzing them.

To identify the large volumetric attack campaigns, we sampled clusters containing a high number of requests or high number of unique usernames. At U1, we found eight such clusters with $\mathsf{NR} \geq 1,000 \vee \mathsf{NU} \geq 1,000$. At U2, we sampled 12 clusters with $\mathsf{NR} \geq 5,000 \vee \mathsf{NU} \geq 5,000$. We chose these thresholds by manually observing that the clusters found after these thresholds do not clearly show malicious behavior. As our goal is to characterize attacks, we focus on precise identification of attack campaigns, foregoing high recall. Thus we sorted possible attacks by a few different metrics. For example, we sorted by the number of unique usernames and found one additional attack at U1; all the top clusters at U2 had already been found using the volumetric thresholds.

All of the above sample clusters did not exhibit any targeted behavior—sending roughly one unique password to each user on average. Therefore, to capture attack campaigns exhibiting targeted behavior, we consider clusters sending a high number of unique passwords submitted per username on average. Although at U1 we did not find any such clusters appearing to be attacks, we discovered eight clusters from U2 that submitted an average of at least 25 unique passwords per user. Altogether, we sampled nine attack clusters at U1 and 20 attack clusters at U2. Attack clusters at U1 sent on average 8,432 requests to 1,294 unique usernames, with a total number of 41 successful logins among these 9 clusters. At U2, we saw an average of 14,358 requests submitted per cluster to 7,614 unique usernames, with a total number of 1,116 successful logins among all 20 clusters. We describe these clusters further in Section 6.1.

6 Analysis of Attack Campaigns

As seen in the last section, Araña's FCA pipeline helped us identify 29 clusters that are probable attack campaigns. We first describe a subset of these campaigns in more detail, to better understand attack behavior as seen in practice. In Section 6.2 we generalize from these case studies to identify a variety of observed higher-level attacker behaviors.

6.1 Example Attack Campaigns

First we describe some attack clusters representative of different types of attack behavior we observed and show how we group some of them into campaigns involving more than one cluster. We show the timeline of the attack campaigns discussed in Figure 7, and the full list of attack clusters we found is shown in Figure 8.

Previously reported attacks. Three attacks were manually identified and discussed in prior work [7]. All three were also detected via our FCA pipeline: Attack #1 from [7] corresponds to Clusters 1 & 6, Attack #2 corresponds to Cluster 2, and Attack #3 corresponds to Cluster 10 (also shown in Figure 7).

Attack #1 from [7] was a credential stuffing attack distributed over four IPs that were active on different days. Each of the IPs submitted lots of requests very quickly, with a peak rate of over 100 requests per second. As such, these were easily identified manually as related attacks by the similar attack behaviors and time period when they were active. This still required significant manual analysis of Gossamer logs; however, our FCA approach automates this analysis. Two clusters identified by the FCA method capture the bulk of the $\mathcal{L}$ sets associated with these attacks. One $\mathcal{L}$ set containing 16,035 requests that were previously manually identified was omitted; this $\mathcal{L}$ set had a lower failure rate (0.67) than our heuristic filtering thresholds (which was 0.77 at U1).

Attack #2 from [7] was a high volume credential stuffing attack using SentryMBA [42] to send requests from a single source IP address. Since the attack consisted of only one IP address on a single date, the cluster consisted of one $\mathcal{L}$ set, ranked second by volume of requests in our FCA approach.

Attack #3 from [7] was observed at U2 and involved 12 distinct IP addresses with an average rate of 188 requests per second. This attack was easy to detect due to its sheer volume and rate of requests. Manual inspection revealed that the attack requests mimicked SMTP and IMAP clients, which helped in manually clustering them. Again, our FCA pipeline automated this step, placing all activity from these IP addresses into a single attack cluster.

Our new FCA approach helps automatically identify these attack campaigns with little error relative to the manual analysis used in prior work [7]. However, it did split the first attack into two clusters, and it also missed an IP address from that attack. Across these three attack campaigns, 17 $\mathcal{L}$ sets were identified by both the manual and FCA approach as being part of attack campaigns, and one $\mathcal{L}$ set was identified by the manual approach but not the FCA approach. We show the corresponding confusion matrices in Appendix D. Note that the FCA approach also found many attack campaigns hard to manually detect that we describe next.

Future work can look into whether unsupervised clustering

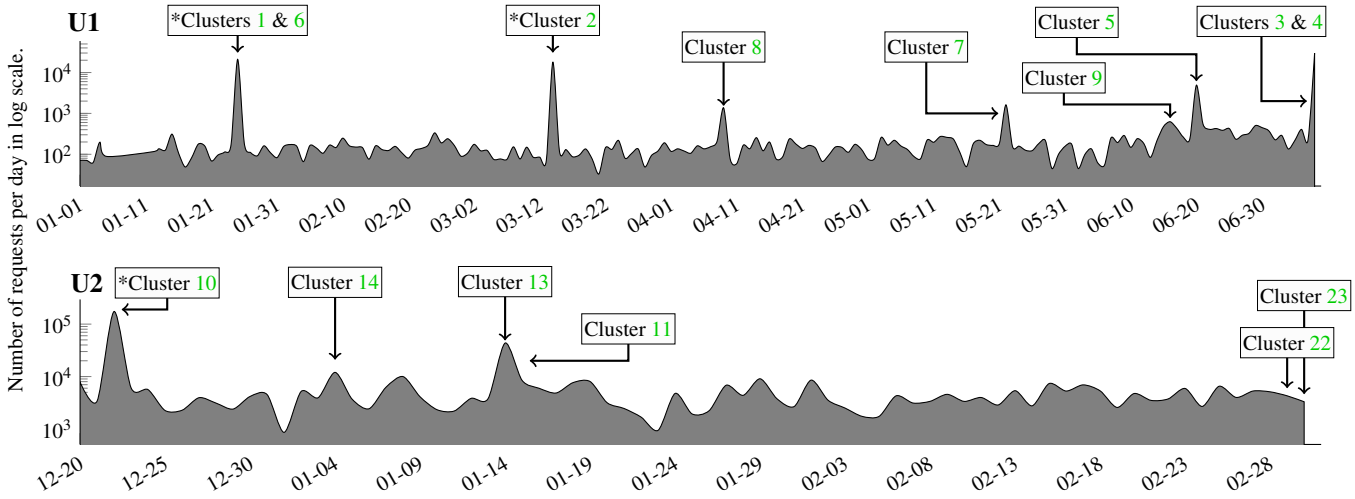

Figure 7: Number of requests per day for the 1,752 and 6,408 potentially malicious $\mathcal{L}$ sets at U1 and U2 respectively as shown in Figure 5. Attack clusters we found in Section 4 are shown in red boxes. Clusters marked with a * were identified in prior work [7] as high volume attacks. Note that the x- and y-axes are different for the two universities for better visualization.

approaches can be improved to be more accurate than manual inspection. For now it is clear that FCA helps find and characterize attack campaigns in an automated fashion that can later be investigated by an analyst. We now discuss in further detail the new attacks discovered using our pipeline.

Clusters 5 & 7: Repeated attacks from the same IP. We saw two credential stuffing attacks at U1 (Clusters 5 & 7) a month apart (May 22 and June 20) from the same IP address and user agent (and thus, likely from the same attacker) attempting to login to 4,480 and 1,347 users respectively. The five user agents used in the June 20 attack were a subset of the seven used in the May 22 attack, and the two attacks targeted 295 overlapping usernames. Thus we believe that the IP was under the same attacker's control for both attacks. This IP was active on 24 other days, but only submitted between one and four requests per day to nine distinct usernames over that time period. We believe the attack exhibited credential stuffing behavior, as the fractions of breached passwords submitted were 0.63 and 0.32 respectively, and less than 8% of the passwords submitted were weak passwords. Other features such as the AUP, FTP, and FSP were relatively similar between the two clusters. All but two usernames were incorrect across the clusters, so we believe the attack was curated for U1.

Cluster 12: Multi-day attack from a single IP. Although most of the attacks we saw were finished within a 24 hour period, some attackers may spread out the attack over multiple days in an attempt to stealthily avoid detection. We saw this behavior in Cluster 12 at U2. In this attack, one IP from Microsoft Corporation ISP sent one request per minute on 15 days in a two month period. In total, the IP sent 13,289 requests to 8,192 unique usernames. There was evidence of credential stuffing, as 61% of passwords submitted appeared in breach data known to Gossamer, and fewer than 0.04% of them were weak passwords. We did not see any evidence of targeted behavior, as there was an average of only one unique password submitted per username; however 63% of the attempted usernames were valid, meaning the attacker curated their password guessing attack for U2. The IP attacked the basic authentication protocol that does not have two factor authentication set up at U2. This IP successfully guessed the passwords of 501 user accounts over this time period, only 163 of which were detected by the security engineers.

We saw similar multi-day attacks in Clusters 15, 18, 19, and 29, albeit for shorter time periods. Such attacks may easily avoid simple volumetric detection methods (as they were missed in the prior work) and thus show the utility of a richer feature-based clustering approach.

Cluster 14: High volume, distributed, credential stuffing. We saw several cases exhibiting high volume, distributed, curated credential stuffing. For example, in Cluster 14 at U2, 843 IPs belonging to 13 ISPs submitted 10,535 requests to 7,771 usernames over the course of 23 hours. Since each IP submitted only around 12 requests, it avoided detection by the manual analysis performed in the prior work [7]. This attack exhibited credential stuffing, with 66% of passwords present in breach data; and 48% of the submitted request contained a valid usernames from U2. During this attack, the involved IPs successfully guessed the passwords of 258 unique user accounts, 74 of which were confirmed by security engineers independently. Clusters 3, 4, 8, 9, 11-13, and 15-19 showed similar signs of high volume, distributed, credential stuffing with varying levels of curation to the target university.

Cluster 17: Possible credential stuffing with unknown breach data. A few attacks showed a high fraction of tweaked passwords (passwords that are a close variant of a breached password for the targeted user), but a lower fraction

ID / Univ.	Duration	# IPs	# ISPs	# unique users (NU)	# requests (NR)	Avg. uniq. pwds per user (AUP)	Frac. weak pwds (FWP)	Frac. pwd in breach (FPIB)	Frac. username, pwd pair in breach (FCIB)	Frac. username in breac (FUIB)	Frac. tweaked pwds (FTP)	Frac. valid user-names (FVU)	Failure rate (FF)	Comp. users	Comp. users flagged.*
1§ / U1	5 h	3	3	10,424	18,093	1.21	0.12	0.56	0.41	0.69	0.06	0.94	1.00	1	1
2§ / U1	4 h	1	1	15,209	17,827	1.10	0.04	0.56	0.22	0.42	0.06	0.97	1.00	14	13
3‡ / U1	14 h	555	4	12,659	15,117	1.00	0.01	0.14	0.02	0.58	0.46	0.78	1.00	14	12
4‡ / U1	11 h	77	2	12,318	14,603	1.00	0.01	0.14	0.02	0.58	0.45	0.78	1.00	7	6
5 / U1	41 m	1	1	4,480	4,593	1.03	0.08	0.63	0.21	0.44	0.08	0.97	1.00	1	1
6§ / U1	4 h	3	1	2,101	2,683	1.21	0.07	0.66	0.48	0.69	0.04	0.98	1.00	0	0
7 / U1	9 h	1	1	1,347	1,481	1.00	0.00	0.32	0.10	0.22	0.05	0.99	1.00	2	2
8 / U1	19 h	85	13	894	1,246	1.00	0.01	0.33	0.08	0.92	0.84	0.92	1.00	2	2
9 / U1	37m	30	7	219	241	1.00	0.06	0.92	0.78	0.96	0.10	0.93	1.00	0	0
10§ / U2	11 h	12	1	76,321	169,573	1.01	0.00	0.03	0.00	0.00	0.00	0.00	1.00	0	0
11 / U2	10 m	663	10	27,488	33,304	1.00	0.10	0.79	0.00	0.00	0.00	0.01	1.00	0	0
12 / U2	15d	1	1	8,192	13,289	1.02	0.04	0.61	0.07	0.54	0.02	0.63	0.93	501	6
13 / U2	3 d	1	1	7,939	12,240	1.30	0.05	0.66	0.10	0.75	0.00	0.35	0.99	120	0
14 / U2	23h	843	13	7,771	10,535	1.01	0.08	0.66	0.00	0.71	0.00	0.48	0.97	258	0
15 / U2	5 d	1	1	4,662	9,714	1.07	0.09	0.84	0.00	0.91	0.00	0.40	0.99	32	0
16 / U2	2 d	2	1	4,934	7,323	1.00	0.06	0.58	0.00	0.23	0.00	0.81	0.84	786	0
17 / U2	2d	3	2	2,513	6,290	1.55	0.03	0.40	0.18	0.91	0.70	0.37	0.99	35	0
18 / U2	10 d	1	1	1,902	5,434	1.37	0.04	0.55	0.27	0.82	0.41	0.39	0.99	13	0
19 / U2	5 d	1	1	3,584	5,261	1.04	0.07	0.76	0.80	0.98	0.18	0.42	1.00	6	0
20 / U2	2 d	3	2	1,756	5,199	1.56	0.04	0.50	0.22	0.92	0.72	0.35	0.98	83	0
21 / U2	23 h	1	1	5,076	5,103	1.00	0.06	0.59	0.00	0.26	0.00	0.80	0.85	777	0
22 / U2	13 h	73	43	75	1,878	25.04	0.82	0.88	0.01	0.31	0.29	0.96	1.00	1	0
23 / U2	21 h	52	38	52	1,296	24.92	0.82	0.89	0.01	0.34	0.32	0.88	1.00	1	0
24 / U2	12 h	4	4	4	101	25.25	0.80	0.86	0.00	0.55	0.50	0.51	0.99	1	0
25 / U2	16 h	3	3	3	80	26.67	0.81	0.75	0.01	0.65	0.58	0.30	0.99	1	0
26 / U2	8 m	1	1	1	27	27.00	0.85	0.85	0.00	0.00	0.00	0.00	1.00	0	0
27 / U2	8 m	1	1	1	27	27.00	0.85	0.78	0.04	1.00	0.96	1.00	1.00	0	0
28 / U2	13 m	1	1	1	25	25.00	0.84	0.88	0.00	0.00	0.00	0.00	1.00	0	0
29 / U2	4 d	1	1	3	468	38.75	0.15	0.59	0.00	0.67	0.16	0.33	1.00	0	0

‡ Since the measurement collection ended during these clusters, some of these measurements (such as NR and NU) may be lower bounds.
§ These clusters are part of campaigns that were discussed in previous work [7].
† Duration units are days (d), hours (h), and minutes (m).
* The last column represents the accounts that were flagged independently by security engineers at the respective universities.

Figure 8: Attack clusters detected using a set of heuristics and manual review. The first column notes the ID of each cluster as we refer to them in the paper. The attack IDs we describe in detail (Section 6) are shown in bold font. We discovered a total of 41 and 1,116 unique compromised users at U1 and U2 respectively.

of breached passwords. For example, in Cluster 17, we saw three IPs submit 6,290 requests to 2,513 unique usernames over the course of two days. Although 91% of the submitted usernames in this attack campaign are present in the breach data used by Gossamer, only 18% of the submitted passwords appear with those usernames. Of the remaining 82% submitted passwords for those users, we that found 70% are only small tweaks of the password(s) present in the breach data with the corresponding username, and 40% of the submitted passwords exactly match the breach data. This may indicate that the attacker is using a breach dataset that is unknown to Gossamer; but because users choose similar passwords across web services [14], we can still detect these breached passwords not present in Gossamer's breach data.

This attack produced successful logins to 35 users, two of which were independently detected by the security engineers. Cluster 20 showed similar signs of possible credential stuffing with unknown breach data, with a high fraction of tweaked passwords. Neither of these attacks exhibited targeted behavior (having AUPs of 1.55 and 1.56 respectively); therefore, we do not believe these are credential tweaking attacks [36].

Clusters 22 & 23: Targeted attacks. In a few cases, we saw a higher average number of unique passwords tried per username. For example, in Cluster 22, we saw 73 IPs submit 1,878 requests to 75 unique usernames, with an average of 25 unique passwords tried per username, each attacking only one user. Cluster 23 was active on the same day and executed a very similar attack: 52 IPs submitted 1,296 requests to 52 usernames, each submitting an average of 25 unique passwords to one unique username, just like Cluster 22. Thus we believe they are part of the same campaign.

In this attack campaign, the attacker clearly used a popular password dictionary, as 82% of passwords submitted had

weak passwords (zxcvbn score of 0). The fraction of valid usernames for Clusters 22 and 23 were 0.96 and 0.88 respectively, indicating that the attack data was probably curated to the university. The campaign was successful in guessing the passwords for two usernames, neither of which was detected by the security engineers.

Clusters 24-28 exhibited similar targeted dictionary attack behavior, although with fewer IPs and a lower number of total requests. Our clustering approach, however, failed to cluster all of these $\mathcal{L}$ sets together. This is a common limitation in agglomerative clustering [41].

6.2 Higher-Level Attack Behaviors Observed

Attack campaigns employ various strategies to maximize their success in compromising user accounts. Broadly, there are two components in an attack campaign that the attacker has to choose: (a) the types of username-password pairs to submit, and (b) how login requests with those username-password pairs are delivered to the target service. We noticed different approaches to each of the two components in the attack campaigns we found at U1 and U2, as we discuss in more detail below. The list of different behaviors we observed as well as some example clusters showing that behavior are shown in Figure 9. We also explore the geographical distribution of attacks, but relegate details to Appendix E.

Types of submitted username-password pairs. A key component in an attack is the set of guessed username-password pairs, as the success of the attack depends on it. There are different strategies for picking username-password pairs—e.g., an attacker can choose usernames belonging to the university and try several popular passwords against those users, or an attacker can choose breached username-password pairs with or without filtering the usernames specific to the university.

Curates usernames to the target university. We found that all attacks at U1 curated their set of usernames, with more than 75% of requests containing a username present in U1. However, at U2, only 7 (out of 20) attack clusters contained more than 50% valid usernames. Clusters 10 and 11 combined submitted nearly 200 K requests, but only 286 of them contained a username present at U2. Even when an attacker attempted to log in multiple times for a particular username, we found that in several cases the username did not exist. Some attacks are therefore rather indiscriminate, trying arbitrary usernames without checking first whether they are valid for the target authentication service.

Targets certain users. An attacker may submit requests against one, a few, or many unique usernames. Among attacks we detected, it was more common for an attack to target a large number of usernames in a "horizontal attack." Most of the attacks we found were horizontal attacks—trying one or two passwords per username but for a large number of usernames. An attack submitting multiple unique passwords against fewer usernames may be exhibiting *targeted* behavior. We found six attack clusters targeting 137 users in total, each with more than 25 unique popular passwords. We do not know if the set of passwords used were the same for different users, as Gossamer logs at U2 do not allow comparing passwords submitted to different usernames.

Attack component	Behavior	Ex. clusters
Type of username-password pairs	Curates to the target university	2, 21
	Targets certain users	22, 23
	Uses breach data	9, 19
	Submits popular passwords	22, 23
Delivery of requests	Distributed among multiple IPs	3, 11
	Distributed among multiple days	12, 19
	Ends quickly (within 24 hours)	1, 11
	Exhibits low interarrival time	5, 11

Figure 9: Different attack behaviors we observed.

Uses breach data. In our analysis, we found that attackers often use prior breaches to source their username-password pairs in what is popularly known as a credential stuffing attack. In six out of nine clusters at U1 and in one cluster at U2, 50% of the submitted username-password pairs are present in the breach data used by Gossamer. Among the remaining 22 clusters, eight clusters (all at U2) had more than 50% of the targeted usernames in a known breach, and all but one attack cluster at U2 submitted passwords that were found in prior breaches. This reiterates the threat of credential stuffing.

Uses popular passwords. Although most attacks used breached passwords, some attacks relied on especially popular passwords. Attackers may use popular password dictionaries [49] curated from prior password breaches for such attacks, and such passwords will, by definition, also be flagged as having appeared in a known breach. We found a number of attack clusters submitting popular weak passwords, such as Clusters 22 & 23. More than 81% of the passwords submitted in these attacks appear in the 1000 most frequent passwords found in the breach data known to Gossamer. Notably, all of these attacks were targeted, submitting more than 25 passwords per user. Both universities have password selection policies that aim to disallow popular, weak passwords. Nevertheless, we found that such attacks were successful in compromising two users at U2.

Delivery methods for attack requests. After choosing the set of username-password pairs to submit, the attacker must decide how to submit them to the target service. Primarily, the attacker must identify how quickly they can submit all the username-password pairs without being detected. To do so, attackers can use multiple IP addresses to parallelize the attacks, and/or spread the attack over a long period of time.

Distributed among multiple IPs. We found several attack campaigns that distributed their login requests over multiple IP addresses. In particular, we identified three large attack clusters using more than 500 IP addresses and five other

clusters each using more than 50 IP addresses. In the two clusters with the most IP addresses at U2 (Clusters 11 & 14), the majority of IPs were flagged as proxies by Blackbox API [2]. By distributing over multiple IPs, an attacker can achieve a higher throughput, as exhibited in cluster 11 which used multiple IPs to achieve a very fast request rate. This distribution of requests across IPs can help to avoid volumetric detection mechanisms, as exhibited in clusters 22 and 23.

Duration of the attack. Attacks can be short-lived and bursty or spread across multiple days. All attack clusters at U1 were short-lived, finishing within 19 hours and four of them completing within five hours. At U2, however, we found that most high volume attacks (submitting more than 5 K requests) span multiple days, and two clusters were active for over 10 and 15 days, respectively. Interestingly, these attacks would be very difficult to detect by looking at their behavior on only one day; however, our clustering approach helps to see the full attack by combining the attack behaviors from multiple days.

Interarrival time. Finally, sending requests too quickly could also trigger alarm or lockout. Therefore, some attackers try to submit requests at a lower rate. However, we see at least four attack campaigns that submitted as much as 60 requests per minute at U1 and U2. These attacks are not curated, meaning that the majority of the submitted usernames do not belong to the target university and thus could not result in a successful login. Both U1 and U2 have a soft rate limiting policy, meaning that too many unsuccessful attempts against a username could lead to account lockout for 15–30 minutes. However, we did not find any attack that submitted requests fast enough to a single user that could trigger that lockout. We are unsure if attackers adapted their attacks to this lockout constraint or if their typical behavior avoids such lockouts.

Endpoints targeted. An attacker may choose an attack endpoint with the least safeguards in place. We found that most of the attacks at U2 targeted the Microsoft basic email authentication endpoint. We suspect that there are four reasons for this high usage of the basic authentication endpoint: (a) this endpoint does not support two factor authentication; (b) it has poor (non-existent) rate limiting of requests, as basic authentication requires frequent submission of passwords; (c) the requesting IP seems to be from Microsoft (as seen by the authentication server); and (d) attackers would have to make only a small change (the URL) to attack different organizations using basic authentication.

7 Discussion

We design Araña, an attack analysis system based on our FCA—filter, cluster, analyze—framework. Using Araña with the logs created by our prior work [7], we identified several attack campaigns in two university login systems. These attack campaigns are spread across 29 clusters as detailed in Figure 8 and compromised 1,157 users across two universities. Although our study is limited to the attacks received by two universities, we believe the patterns we observed across attacks can form a basis for building future defenses against password guessing attacks.

At U1, the most common type of attack we saw was a combination of a high volume, distributed, curated, short-lived, credential stuffing attack; the interarrival times of the attacks varied from 55 milliseconds to 108 seconds. At U2, almost all the attacks were high volume credential stuffing attacks, but they varied in whether they were distributed, curated to the university, short-lived, or exhibiting a low interarrival time. Of the eight lower volume attacks we found, all were curated, short-lived, targeted credential stuffing; and they varied as to whether they were distributed across IPs. The distributions of attacks observed in the two universities are quite different, possibly because Gossamer gathered data from all login endpoints at U2, but only the main login endpoint at U1. Thus, attackers utilizing different endpoints at U1 are not reflected in Gossamer logs.

Efficacy of breach alerting services. To determine how much a breach alerting service could mitigate password guessing attacks, we investigated the characteristics of the password from the last successful login before a user was reported as compromised. We found that 2% of compromise reports were associated with a breached username-password pair (that is, the last successful login to that username before the compromise report was a breached username-password pair), 12% were associated with at least a breached password, and 19% were associated with a tweaked password. At U2, we observed that 11% of compromised users' last logins were made with a breached username-password pair, 46% were made with at least a breached password, and 1.51% were made with a tweaked password. Thus automatically resetting passwords of users using vulnerable, breached, or tweaked credentials could have prevented a significant fraction of account compromises. At U1, 71% of eventually compromised users used a breach password at some point during the instrumentation period. This further underpins the need for proactive breach alerting [27] services that could have saved 47% of account takeovers as stated earlier.

Key observations from attacks. Our findings reiterate the ongoing threat of credential stuffing, which has been the most prevalent and successful form of account compromise attack. We also observed a few low volume targeted attacks against specific users. Attackers often use multiple IPs from cloud providers, VPNs, and network proxies to distribute and hide their attacks, but our FCA approach can identify such campaigns even when each individual IP makes only a handful of requests. Such observations should be taken into account while building defense policies. For example, locking user accounts due to a small number of incorrect attempts rarely translates to higher security, whereas discouraging users from

reusing passwords from other websites and using breach alerting services can be very effective. Proactive breach alerting [27] using services such as HIBP [22] would be very helpful in combating credential stuffing attacks.

Using an FCA approach for attack analysis. Our FCA approach helps analyze attack campaigns by clustering $\mathcal{L}$ sets with similar attack behavior. This enables a security engineer to look at the whole attack, instead of considering login activities from a single IP or on a single day. As we show, several attacks are spread across multiple days and use multiple IPs; in some cases, they may use only one IP and spread the attack over multiple days, making them very hard to detect. We believe clustering seemingly unclear behaviors into groups can help security engineers see the attack pattern, detect hard-to-detect attacks, and have confidence in their judgment. We envision that our FCA approach can be used on authentication logs such as the ones produced by Gossamer [7] to group possible attack campaigns and sort by those that are more likely to be actual attacks. Then analysts can manually investigate further, using the features and groupings produced by the clustering to inform their decision. Throughout the design of our FCA approach, we focused on minimizing false positives by choosing aggressive thresholds and evaluating threshold effects so that an analyst can have confidence that clusters found by Araña are highly likely to be attacks. Thus our FCA approach could reduce the effort needed to analyze the complete logs of all IPs; such a tool could prepare daily (or weekly) reports about potential attacks.

8 Real-World Deployment Constraints

Despite the many campaigns detected by Araña, there are certain challenges that may occur during deployment in real-world settings. Here we talk about a few that we have faced.

Recording password-derived features. Most organizations do not record password-based features for good reason, and while Gossamer [7] showed how to record them safely, it is still an open question whether widespread deployment of recording password-derived features is wise or necessary. Araña provides evidence that certain password-derived features can be useful for attack detection. For example, we identified that password breach statistics are very effective at detecting credential stuffing attacks. On the other hand, password similarity statistics—which require storing passwords in memory for 24 hours—are not as effective for attack detection in the FCA approach. So organizations could log a small set of already proven-helpful password-derived features and apply clustering using those features. We detail clustering results without password based features in Appendix C and leave the open question of which features are the most helpful for detecting different types of attacks to future work.

Reporting compromised accounts. Based on the attack campaigns discovered by our FCA approach, we believe 41 unique user accounts were compromised at U1, and 1,116 were compromised at U2 (some accounts were compromised multiple times by different campaigns).

At U1, 37 of the 41 compromised accounts detected by Araña were already detected via other mechanisms, and manually recorded by the security engineers as compromised. We reported the remaining accounts and received feedback that they were indeed compromised, and the remaining account had already been deleted.

At U2, only six out of 1,116 compromised users detected by Araña had been independently flagged by the security engineers. We reported the rest to U2's security engineers in two rounds. In the first round, we reported 823 compromised accounts, 373 (44%) of which the security engineers confirmed as definitely compromised. For the remaining 450 accounts, U2's security engineers said they could not verify fully due to unavailability of adequate logs but that their best guess was that they were compromised based on what logs were available. In the second round, we reported another 293 compromised usernames; however, we received a similar response mentioning that the IT department did not have adequate logs to determine whether these were compromised.

In practice, confirming if accounts are compromised is nuanced and often relies on indirect indicators like an account being used to send spam or a report from an account owner. Elsewhere, compromise status can be inherently ambiguous. Even so, our experience working with the security engineers suggests that more detailed and persistent logging may help. Likewise, our experience with Araña suggests that password-derived measurements can be helpful to analysts when attempting to characterize and confirm compromises. We also note that this difficulty in confirming account compromise makes it difficult to automatically finetune many of the hyperparameters in Araña (e.g., distance thresholds and percentile thresholds for filtering $\mathcal{L}$ sets). Therefore we had to rely on experimentation with different thresholds and manual analysis on password-derived and volumetric-based measurements which took comparatively much more effort. Nevertheless, Araña provides a way to use Gossamer logs to identify high-volume and distributed attacks that were not detected by existing mechanisms.

9 Conclusion

Using data collected at two universities from a measurement framework for logging password-derived information behavior, we designed a set of features that describe an IP address on a given date, along with a clustering algorithm to group IP addresses active on certain dates together into probable attack campaigns. We describe several of these clusters in full detail to show the differences and similarities between attacks, and we discuss our observations about behavioral patterns of the attacks as a whole.

Our results indicate that clustering approaches can aid an analyst in detecting and labeling suspicious groups of requests

that may be part of the same attack campaign. They also provide initial progress towads the future design and deployment of real-time, automated attack campaign detection tools.

Acknowledgments

We thank the IT engineers, especially Dan Villanti, Jerry Shipman, Bridget Bartell, Ryan Larschedit, Bruce LaBuda, and the IT leadership at both universities for their support in the research. We also thank the anonymous reviewers and our shepherd for their insightful feedbacks and improving the paper significantly. This work was supported in part by the University of Wisconsin–Madison Office of the Vice Chancellor for Research and Graduate Education with funding from the Wisconsin Alumni Research Foundation and NSF grant CNS #1763810. Mazharul Islam was supported in part through the US Department of Commerce award #70NANB21H043.

References

[1] https://github.com/islamazhar/Arana-Public. 2

[2] https://blackbox.ipinfo.app. 13

[3] UA-parser. https://github.com/ua-parser/uap-python. 8

[4] ABBOTT, J., AND PATIL, S. How Mandatory Second Factor Affects the Authentication User Experience. In *2020 CHI Conference on Human Factors in Computing Systems* (New York, NY, USA, 2020), CHI '20, Association for Computing Machinery, p. 1–13. 3

[5] AYSA, M. H., IBRAHIM, A. A., AND MOHAMMED, A. H. IoT DDOS Attack Detection Using Machine Learning. In *2020 4th International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT)* (2020). 3

[6] BHARDWAJ, A. Silhouette coefficient. https://towardsdatascience.com/silhouette-coefficient-validating-clustering-techniques-e976bb81d10c, 2020. 8, 17

[7] BOHUK, M. S., ISLAM, M., AHMAD, S. S., SWIFT, M., RISTENPART, T., AND CHATTERJEE, R. Gossamer: Securely Measuring Password-based Logins. In *USENIX Security 22* (2022), USENIX Association. 1, 2, 3, 4, 5, 9, 10, 11, 13, 14, 18

[8] BONNEAU, J., HERLEY, C., VAN OORSCHOT, P. C., AND STAJANO, F. The quest to replace passwords: A framework for comparative evaluation of web authentication schemes. In *2012 IEEE Symposium on Security and Privacy* (2012). 2

[9] BONNEAU, J., HERLEY, C., VAN OORSCHOT, P. C., AND STAJANO, F. Passwords and the evolution of imperfect authentication. *Communications of the ACM 58*, 7 (2015). 3

[10] BUCZAK, A. L., AND GUVEN, E. A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection. *IEEE Communications Surveys & Tutorials 18*, 2 (2016). 3

[11] BUITINCK, L., LOUPPE, G., BLONDEL, M., PEDREGOSA, F., MUELLER, A., GRISEL, O., NICULAE, V., PRETTENHOFER, P., GRAMFORT, A., GROBLER, J., LAYTON, R., VANDERPLAS, J., JOLY, A., HOLT, B., AND VAROQUAUX, G. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning* (2013), pp. 108–122. 9

[12] CHATTERJEE, R., ATHAYLE, A., AKHAWE, D., JUELS, A., AND RISTENPART, T. pASSWORD tYPOS and how to correct them securely. In *2016 IEEE Symposium on Security and Privacy* (2016). 2

[13] COLNAGO, J., DEVLIN, S., OATES, M., SWOOPES, C., BAUER, L., CRANOR, L., AND CHRISTIN, N. "It's not actually that horrible": Exploring Adoption of Two-Factor Authentication at a University. In *2018 CHI Conference on Human Factors in Computing Systems (CHI '18)* (2018), Association for Computing Machinery. 3

[14] DAS, A., BONNEAU, J., CAESAR, M., BORISOV, N., AND WANG, X. The Tangled Web of Password Reuse. In *Network and Distributed Systems Security (NDSS) Symposium 2014* (2014), vol. 14. 2, 3, 11

[15] DEIGHTON, K. Tech Companies Push Users to Adopt Two-Factor Authentication. https://www.wsj.com/articles/tech-companies-push-users-to-adopt-two-factor-authentication-11635807088, 2021. 3

[16] DOERFLER, P., THOMAS, K., MARINCENKO, M., RANIERI, J., JIANG, Y., MOSCICKI, A., AND MCCOY, D. Evaluating Login Challenges as a Defense Against Account Takeover. In *28th International Conference on World Wide Web (WWW '19)* (2019). 3

[17] FLORENCIO, D., AND HERLEY, C. A Large-Scale Study of Web Password Habits. In *16th International Conference on World Wide Web (WWW '07)* (2007), Association for Computing Machinery. 2

[18] FREEMAN, D., JAIN, S., DUERMUTH, M., BIGGIO, B., AND GIACINTO, G. Who Are You? A Statistical Approach to Measuring User Authenticity. In *Network and Distributed Systems Security (NDSS) Symposium 2016* (2016). 1, 3, 5, 6, 8

[19] HERLEY, C., AND SCHECHTER, S. E. Distinguishing Attacks from Legitimate Authentication Traffic at Scale. In *Network and Distributed Systems Security (NDSS) Symposium 2019* (2019). 1, 6

[20] HITAJ, B., GASTI, P., ATENIESE, G., AND PEREZ-CRUZ, F. PassGAN: A Deep Learning Approach for Password Guessing. In *International conference on applied cryptography and network security* (2019), Springer, pp. 217–237. 3

[21] HO, G., SHARMA, A., JAVED, M., PAXSON, V., AND WAGNER, D. Detecting Credential Spearphishing Attacks in Enterprise Settings. In *USENIX Security 17* (2017), USENIX Association. 6, 16

[22] HUNT, T. Have I Been Pwned? https://haveibeenpwned.com/Passwords/, 2018. 2, 3, 13

[23] JUBUR, M., SHRESTHA, P., SAXENA, N., AND PRAKASH, J. Bypassing push-based second factor and passwordless authentication with human-indistinguishable notifications. In *2021 ACM Asia Conference on Computer and Communications Security* (New York, NY, USA, 2021), ASIA CCS '21, Association for Computing Machinery, p. 447–461. 3

[24] KELLEY, P. G., KOMANDURI, S., MAZUREK, M. L., SHAY, R., VIDAS, T., BAUER, L., CHRISTIN, N., CRANOR, L. F., AND LOPEZ, J. Guess Again (and Again and Again): Measuring Password Strength by Simulating Password-Cracking Algorithms. In *2012 IEEE Symposium on Security and Privacy* (2012). 2

[25] KOMANDURI, S. Modeling the Adversary to Evaluate Password Strength With Limited Samples. https://kilthub.cmu.edu/articles/thesis/Modeling_the_Adversary_to_Evaluate_Password_Strength_With_Limited_Samples/6720701/1, 2018. 3

[26] KOMANDURI, S., SHAY, R., KELLEY, P. G., MAZUREK, M. L., BAUER, L., CHRISTIN, N., CRANOR, L. F., AND EGELMAN, S. Of Passwords and People: Measuring the Effect of Password-Composition Policies. In *SIGCHI Conference on Human Factors in Computing Systems (CHI '11)* (2011), Association for Computing Machinery. 2

[27] LI, L., PAL, B., ALI, J., SULLIVAN, N., CHATTERJEE, R., AND RISTENPART, T. Protocols for Checking Compromised Credentials. In *2019 ACM SIGSAC Conference on Computer and Communications Security (CCS '19)* (2019), Association for Computing Machinery. 2, 13

[28] MA, J., YANG, W., LUO, M., AND LI, N. A study of probabilistic password models. In *2014 IEEE Symposium on Security and Privacy* (2014). 3

[29] MAXMIND. GeoIP2 Database. https://www.maxmind.com/en/geoip2-databases. 5, 8

[30] MAZUREK, M. L., KOMANDURI, S., VIDAS, T., BAUER, L., CHRISTIN, N., CRANOR, L. F., KELLEY, P. G., SHAY, R., AND UR, B. Measuring Password Guessability for an Entire University. In *2013 ACM SIGSAC Conference on Computer and Communications Security (CCS '13)* (2013), Association for Computing Machinery. 2

[31] MCINNES, L., HEALY, J., AND ASTELS, S. HDBSCAN: Hierarchical Density-Based Clustering. *The Journal of Open Source Software 2*, 11 (2017). 17

[32] MELICHER, W., UR, B., SEGRETI, S. M., KOMANDURI, S., BAUER, L., CHRISTIN, N., AND CRANOR, L. F. Fast, Lean, and Accurate: Modeling Password Guessability Using Neural Networks. In *USENIX Security 16* (2016), USENIX Association. 3

[33] MÜLLNER, D. Modern hierarchical, agglomerative clustering algorithms. *arXiv preprint arXiv:1109.2378* (2011). 2, 6, 8

[34] NARAYANAN, A., AND SHMATIKOV, V. Fast Dictionary Attacks on Passwords Using Time-Space Tradeoff. In *12th ACM Conference on Computer and Communications Security* (New York, NY, USA, 2005), CCS '05, Association for Computing Machinery, p. 364–372. 3

[35] OPENWALL. John The Ripper. https://www.openwall.com/john/. 2

[36] PAL, B., DANIEL, T., CHATTERJEE, R., AND RISTENPART, T. Beyond credential stuffing: Password similarity models using neural networks. In *2019 IEEE Symposium on Security and Privacy (SP)* (2019). 3, 11

[37] PAL, B., ISLAM, M., BOHUK, M. S., SULLIVAN, N., VALENTA, L., WHALEN, T., WOOD, C., RISTENPART, T., AND CHATTERJEE, R. Might I Get Pwned: A Second Generation Compromised Credential Checking Service. In *USENIX Security 22* (2022), USENIX Association. 3

[38] SATOPAA, V., ALBRECHT, J., IRWIN, D., AND RAGHAVAN, B. Finding a "Kneedle" in a Haystack: Detecting Knee Points in System Behavior. In *2011 31st International Conference on Distributed Computing Systems Workshops* (2011). 8

[39] SCHECHTER, S., TIAN, Y., AND HERLEY, C. StopGuessing: Using Guessed Passwords to Thwart Online Guessing. In *2019 IEEE European Symposium on Security and Privacy (EuroS&P)* (2019). 3, 5, 6

[40] STEUBE, J., AND GRISTINA, G. Hashcat. https://hashcat.net/hashcat/. 2

[41] TAN, P.-N., STEINBACH, M., KARPATNEAND, A., AND KUMAR, V. Introduction to Data Mining, 2nd Edition. *People's Posts and Telecommunications Publishing House, Beijing* (2019). 12

[42] THEE, D. Sentry MBA. https://e.cyberint.com/hubfs/Sentry%20MBA%20Report/CyberInt_Sentry-MBA_Report.pdf. 9

[43] THOMAS, K., LI, F., ZAND, A., BARRETT, J., RANIERI, J., INVERNIZZI, L., MARKOV, Y., COMANESCU, O., ERANTI, V., MOSCICKI, A., MARGOLIS, D., PAXSON, V., AND BURSZTEIN, E. Data Breaches, Phishing, or Malware? Understanding the Risks of Stolen Credentials. In *2017 ACM SIGSAC Conference on Computer and Communications Security (CCS '17)* (2017), Association for Computing Machinery. 1, 2, 3

[44] THOMAS, K., PULLMAN, J., YEO, K., RAGHUNATHAN, A., KELLEY, P. G., INVERNIZZI, L., BENKO, B., PIETRASZEK, T., PATEL, S., BONEH, D., AND BURSZTEIN, E. Protecting Accounts from Credential Stuffing with Password Breach Alerting. In *USENIX Security 19* (2019), USENIX Association. 3

[45] TREMANTE, M. End User Security: Account Takeover Protections with Cloudflare. https://blog.cloudflare.com/account-takeover-protection/, 2021. 2, 3

[46] WANG, D., ZHANG, Z., WANG, P., YAN, J., AND HUANG, X. Targeted online password guessing: An underestimated threat. In *2016 ACM SIGSAC Conference on Computer and Communications Security (CCS '16)* (2016), Association for Computing Machinery. 3

Reason	Count	Percent
Bulk credential testing	1255	69.3%
Simultaneous use from different locales	438	24.2%
Self-reported password compromise	45	2.5%
Blocked for spamming via Office 365	23	1.3%
Password misused	20	1.1%
Spamming via Office 365 / Gmail	11	0.6%
Other (incl. reports from Duo fraud, third-party, etc.)	20	1.1%

Figure 10: The reasons for compromise as noted at U1 and the number of instances of such compromise. Users reported multiple times are counted as distinct instances.

[47] WEIR, M., AGGARWAL, S., MEDEIROS, B. D., AND GLODEK, B. Password Cracking Using Probabilistic Context-Free Grammars. In *2009 30th IEEE Symposium on Security and Privacy* (2009). 2, 3

[48] WIDUP, S., RUDIS, B., HYLENDER, D., AND SPITLER, M. 2015 Data Breach Investigations Report. https://www.researchgate.net/publication/289254638_2015_Verizon_Data_Breach_Investigations_Report, 2015. 1, 3

[49] WIKIPEDIA. List of the most common passwords. https://en.wikipedia.org/wiki/List_of_the_most_common_passwords. 12

A Reasons for Compromise Reports

As described in Section 3, we received the logs of compromised account reports from both universities. We show the breakdown of the reasons for compromise as recorded at U1 in Figure 10. We that found 69% of accounts were reported as compromised due to large-scale automated attacks (referred to as "Bulk credential testing" at U1). For compromised accounts reported within one hour of the time of attack, 46% were classified as "Bulk credential testing", 17% as self-reported password compromise, and the remaining 37% were split between the other 10 reasons appearing in the dataset. For the six compromised accounts that took longer than 300 hours (12.5 days) to report, all were reported as "Simultaneous use from different locales."

At U2, the recorded reasons for compromise are very coarse: 99% are reported simply as "compromised accounts". Three accounts were disabled based on requests from the human resources department (former employee), and two were disabled as the user is "deceased". No username is reported more than once for different reasons.

B Using DAS to Detect Attacks

The DAS algorithm [21] has been used successfully in other contexts, such as detecting spearphishing attacks. Applied to our context, the DAS algorithm takes a set $\mathcal{L}_1, \mathcal{L}_2, \ldots$ and orders the sets as follows. For some configured subset of numerical features, we first associate an ordering operator over possible feature values (e.g., higher failure rate FF is more suspicious). Then we associate to $\mathcal{L}_i$ a score that is equal to the number of other sets $\mathcal{L}_j$ $(i \neq j)$ such that $\mathcal{L}_i$'s features are all strictly more suspicious than $\mathcal{L}_j$'s features. We

can then obtain a partial ordering over the set of $\mathcal{L}$ sets based on these scores.

In finding a configuration of features for DAS, we optimized for the fraction of the top 50 $\mathcal{L}$ sets as ordered by DAS that were associated with a compromised username, as reported by the security engineers. In doing so, we found that using two simple volumetric features—the number of requests submitted (NR) and the number of unique users contacted (NU)—yielded one of the highest fractions of $\mathcal{L}$ sets associated with a compromised username. In fact, computing DAS with the features NU and NR yielded 41 out of the top 50 $\mathcal{L}$ sets and 87 out of the top 100 $\mathcal{L}$ sets associated with a compromised username. Adding as a feature the number of consecutive days an IP has been active (CD) increased it to 50 out of the top 50 and 98 out of the top 100. Thus we hypothesize that these three features are most correlated with the current mechanisms used by the university IT offices for detecting attacks that cause compromise reports.

DAS with NR, NU, and CD may only be useful for discovering attacks already being caught by existing countermeasures, so we explore extending to further features that the IT offices may not be considering. For example, features such as the fraction of passwords in a breach (FPIB), the fraction of tweaked passwords (FTP), and average unique passwords per user (AUP) may all help in detecting attacks. We tried running DAS on a configuration with a much richer feature set: NR, NU, FPIB, FCIB, FTP, AUP, and FF. Despite manually flagging all 50 as probable attacks, only 19 were flagged as attacks in the compromise database. This indicates that the IT offices may be missing many attacks that could be found with a richer combination of features, and it also suggests that the compromise reports may not be a good ground truth.

Our experiments with DAS show that it has promise for discovering attacks. While naively using it with just volumetric features seems to miss more subtle attack behaviors, when used with richer Gossamer-enabled features DAS can even discover (successful) attacks that are not being caught by existing countermeasures. This suggests that future deployments may want to consider using DAS-style approaches for remote guessing attack detection, similar to its original use with spearphishing. However, from the perspective of our goal of better characterizing attack campaigns, DAS has various limitations. In particular, it cannot group IP addresses into attack campaigns, and distributed attacks that use multiple IPs will be treated as separate attacks.

Thus we consider clustering as an unsupervised approach to grouping suspicious IP addresses into attack campaigns.

C Additional Clustering Results

Clustering Quality. Before settling on agglomerative clustering for grouping $\mathcal{L}$ sets into potential campaigns, we tried a number of other clustering techniques. We considered the K-means++, DBSCAN, HDBSCAN, and agglomerative clustering techniques in this study on the same set of features from Figure 2. Since K-means++ cannot work with a custom similarity model, we applied principal component analysis (PCA) on all feature values of $\mathcal{L}$ sets and projected them in a two dimensional space to run K-means++ clustering. We also tried projecting to more than two dimensions but did not observe any noticeable effect on the clustering quality. To judge the quality of the clustering techniques, we used the silhouette score [6], which computes the normalized difference between the average inter-class and intra-class distances. We did a grid search over all hyperparameters for each of the clustering techniques and reported the best silhouette score in Figure 11.

Clustering algorithm	Silhouette score ↑ U1	U2
PCA and K-Means++	−0.13	−0.09
DBSCAN	−0.39	−0.30
HDBSCAN	−0.12	−0.08
Agglomerative	+0.19	+0.17

Figure 11: Silhouette scores of different clustering models.

All clustering methods except agglomerative clustering received a negative silhouette score, signifying poor quality clusters. We hypothesize that the poor performance is because our similarity model is not a metric similar to Euclidean distance, which is essential for K-means++ to produce meaningful clusters. DBSCAN performed well when the clusters were of the same density—that is, when $\mathcal{L}$ sets belonging to the same cluster are uniformly distributed inside a cluster. However, in our use case, it is common to have attack campaigns of different densities. While HDBSCAN [31] is specifically designed to handle non-uniform clusters and produced relatively better clusters for a number of attack campaigns, it still received a lower silhouette score than agglomerative clustering.

Sensitivity of clustering threshold. To analyze how sensitive our clustering results are for different thresholds, we run two experiments by changing (a) the filtering threshold which is applied to $\mathcal{L}$ sets and (b) the distance threshold that dictates whether two clusters would be merged or not. Our findings show that the clustering results are sensitive to changes in the distance threshold but remain relatively the same for changes to filtering thresholds.

For the first experiment, we change the filtering threshold to a less aggressive 70^{th} percentile instead of the 80^{th} percentile as we did originally, while keeping the distance threshold the same as before at 0.51. After manual analysis, we find no noticeable change in the likely malicious clusters in comparison to Araña's clustering results. This indicates that clustering results may not be sensitive to changes in the filtering thresholds.

In the second experiment, in addition to the filtering thresholds, we also change the distance threshold by applying a knee locator method on $\mathcal{L}$ sets filtered by the 70^{th} percentile. This gives a distance threshold of 0.41, which is lower than the original distance threshold of 0.51. Although the majority

of the resulting clusters remain the same, we observe a few noticeable changes in the clusters as we describe below.

We sampled the top 20 (16 untargeted and 4 targeted) likely malicious clusters using the same sampling criteria used in Araña. The first 16 clusters had NR $\geq 5,000 \vee$ NU $\geq 5,000$, exhibiting high volume, untargeted behavior. Out of these 16, we discover that 9 clusters were exactly the same as the ones found in Araña's clustering results. However, we identify five new untargeted clusters which we did not see in Araña's clustering result. Our manual analysis confirms one of them to be a malicious attack campaign, one to be clearly benign behavior, and the other three to contain a mix of benign and malicious behavior. Furthermore, we observe that two attack campaigns which were previously split into five clusters by Araña, are more accurately represented by two distinct clusters with this new distance threshold. The remaining 4 clusters with AUP ≥ 25 exhibited targeted behavior and were identical to those found by Araña.

Lastly, we notice that, with the new relaxed distance threshold, two targeted attack campaigns detected by Araña are completely missed. This is because lowering the distance threshold introduced spurious $\mathcal{L}$ sets that did not exhibit the same targeted behavior as the other malicious $\mathcal{L}$ sets showing clear targeted behavior. Thus, the AUP value of the mixed cluster was reduced below our selection threshold 25.

In conclusion, we find that Araña's clustering results are particularly sensitve to changes to the distance thresholds. While lowering the distance theshold can allow for the discovery of more potential attack campaigns, it also increases the chance of mixing benign and malicious $\mathcal{L}$ sets in the same cluster. To prioritize precision and a low false positive rate, we choose the higher percentile filtering thresholds which result in a high distance threshold, minimizing the chance of benign and malicious $\mathcal{L}$ sets appearing in the same cluster.

Clustering without password-based features. To understand the importance of recording users' password-based information, we reran Araña at U2 without the six password-based features; We found that without these features, we were still able detect 92% (1,570) of 1,709 malicious $\mathcal{L}$ sets and 80% (16) of 20 attack clusters discovered by Araña as shown in Figure 8. However, it missed one multi-day credential stuffing attack from a single IP address (cluster #15) and three targeted attacks (clusters #22, #23, #29). We hypothesize that since all $\mathcal{L}$ sets in attack cluster #15 have similar values across five password guessability features and all $\mathcal{L}$ sets in attack clusters #22, #23, and #29 have similar feature values for AUP, Araña placed them in the same respective attack clusters predicting that they originate from the same attackers even if they were spread across multiple days.

Additionally, clustering without password-based features flagged 46 new $\mathcal{L}$ sets. We manually analyzed these 46 $\mathcal{L}$ sets and suspect that at least four $\mathcal{L}$ sets are malformed clients, since they submitted the same password against a single user. Moreover, we found that $\mathcal{L}$ sets within the same cluster have significantly different values for password-based features. For example, eight $\mathcal{L}$ sets formed a new cluster in which one IP address had an FPIB value of 0, but the other seven $\mathcal{L}$ sets had FPIB > 0.55 with an average FPIB of 0.66. For the remaining 34 $\mathcal{L}$ sets, it is difficult for us to judge them as either fully malicious or benign, since the password-based features varied within a cluster. Thus we believe password-based features are important for grouping attack traffic into attack campaigns, as well as manually judging a cluster as malicious or benign.

Attack # [7]	# $\mathcal{L}$ sets flagged [7]	Clusters	# $\mathcal{L}$ sets flagged by FCA	
			Flagged	Not Flagged
#1 (at U1)	7	1,6	6	1
#2 (at U1)	1	2	1	0
#3 (at U2)	12	10	12	0

Figure 12: Confusion matrices of the number of $\mathcal{L}$ sets corresponding to the three attack campaigns found in prior work [7] using their manual approach versus our FCA approach.

D Comparing Araña to prior work [7]

In Section 6.1, we discuss the three attack campaigns manually identified in prior work [7]. Here we compare in more detail how their attack identification compares to the corresponding clusters found using Araña. In Figure 12, we present three confusion matrices—one for each attack. These show how many $\mathcal{L}$ sets were labeled as part of the attack or not for both methods. For two attacks, agreement was perfect (unsurprising for attack #2 which only emanated from a single IP on a single day), and for attack #1 Araña missed just one $\mathcal{L}$ set. We believe that this false negative arose because that $\mathcal{L}$ set set fell on the next day (after midnight) compared to the prior $\mathcal{L}$ sets, suggesting that there can be some noise introduced by our 24-hour cut-offs for login sets. Even so, an analyst using Araña could, in this case, easily detect the false negative manually since the IP address is the same.

E Geographical Source of Attacks

We use the ISP of a request to determine the country or countries of origin. Since a campaign often contains multiple IPs, there may be more than one country of origin. Figure 13 shows the most common countries from which attacks originated. At U1 and U2, the vast majority of malicious IP addresses originated from within the United States.

Country	# of requests
United States	65,474
Germany	6,170
Morocco	3,467
Canada	152
Brazil	112

Country	# of requests
United States	234,515
Russia	19,003
Ireland	13,130
Netherlands	5,678
Canada	1,571

Figure 13: The five most common countries at U1 (**left**) and U2 (**right**) by number of attack requests.