Transforming Raw Authentication Logs into Interpretable Events

Seth Hastings
The University of Tulsa
seth-hastings@utulsa.edu

Corey Bolger
The University of Tulsa
corey-bolger@utulsa.edu

Philip Shumway
The University of Tulsa
philip-shumway@utulsa.edu

Tyler Moore
The University of Tulsa
tyler-moore@utulsa.edu

***Abstract*—Authentication logs can be helpful to Security Operations Centers (SOCs), but they are often messy, reporting details more relevant to system configurations than user experiences and spreading information on a single authentication session across multiple entries. This paper presents a method for converting raw authentication logs into user-centered "event logs" that exclude non-interactive sessions and capture critical aspects of the authentication experience. This method is demonstrated using real data from a university spanning three semesters. Event construction is presented along with several examples to demonstrate the utility of event logs in the context of a SOC. Authentication success rates are shown to widely vary, with the bottom 5% of users failing more than one third of authentication events. A proactive SOC could utilize such data to assist struggling users. Event logs can also identify persistently locked out users. 2.5% of the population under study was locked out in a given week, indicating that interventions by SOC analysts to reinstate locked-out users could be manageable. A final application of event logs can identify problematic applications with above average authentication failure rates that spike periodically. It also identifies lapsed applications with no successful authentications, which account for over 50% of unique applications in our sample.**

I. Introduction

A Security Operations Center (SOC) serves as the "nerve center" of an organization's cybersecurity efforts. It should receive inputs from multiple sources, be sensitive to stimuli that may signal danger, and present the organization with a comprehensive representation of its environment. The primary functions range from monitoring, assessing, and defending against cyber threats, to surveillance of networks, servers, applications and users. This enables the SOC to identify pain points, potential vulnerabilities, and areas for improvement.

As such, a SOC is heavily limited by the quality of its inputs, i.e., its data sources. Many tools are utilized to develop and leverage data sources, such as Security Information and Event Management (SIEM) systems, Intrusion Detection Systems (IDS), vulnerability management tools, and other analytical tools. These systems work together to enable SOCs to detect, investigate, and respond to issues at speed.

Workshop on SOC Operations and Construction (WOSOC) 2024
1 March 2024, San Diego, CA, USA
ISBN 979-8-9894372-3-8
https://dx.doi.org/10.14722/wosoc.2024.23003
www.ndss-symposium.org

One primary data source for SOCs is authentication logs. Controlling who uses a given service or application, and in what capacity, is key to both proper security and functionality. Many organizations have deployed single sign-on (SSO) services such as Microsoft Azure AD (now Azure Entra AD) to streamline their users' authentication experience.

Currently, authentication logs are used to investigate user and application issues, as well as sources for systems that generate alerts of suspicious activities [13], [9]. For example, SOC analysts can identify potential account takeovers when logs indicate login attempts from an unexpected country or high frequency failures. Increasingly, artificial intelligence (AI) and machine learning (ML) models are employed to flag anomalies, promising to reduce time to detection. However, Zhao et al [14] identify several limitations including difficulty dealing with complex abnormal log patterns, poor interpretability of alerts, and lack of domain knowledge.

Traditional monitoring involves engineers examining logs and writing keyword and regular expression based rules for detection. This method is growing more challenging as the number of components and variety of logs increases, resulting in noisy datasets that require extensive domain knowledge to interpret, with new and updated service components producing ever diversifying log messages. While AI and ML systems can offer sensitivity to abnormality, they struggle with interpretation: engineers might be alerted that a given state is anomalous, but it is unclear why something is an anomaly, and what a "normal" pattern would look like.

Raw authentication logs are noisy. They have not been created with easy interpretability in mind. A single login attempt often generates dozens of log entries, each apparently disconnected from another. Wading through that mess, either manually or with an automated system, can be problematic.

In this paper, we describe a process to construct interpretable, user-centric "event logs" from raw authentication logs that reduce noise, eliminate redundant entries, and combine entries into discrete user experiences. This event-focused dataset can be implemented in several ways: as an input for an IDS that allows for more interpretable alerts, as a more straightforward dataset for investigation that lowers the bar for domain knowledge, and as a means of generating performance metrics that enable proactive identification of struggling users or applications.

The paper is organized as follows. Section II reviews related

work. Section III describes the method for distilling raw authentication logs into distinct events. Section IV explores examples of how event data can be leveraged. Finally, we conclude in Section V.

II. Related Work

Prior work incorporating authentication logs falls into a few broad categories. First, a small group of usability research on multi-factor authentication (MFA), some of which has used authentication logs to measure adoption rates and basic counts of errors associated with MFA use[11] [3] [10]. The most relevant examples being from Reynolds et al., who tracked users through a 90 day MFA adoption period at a university. They introduced "recovery time", defined as the time between a failed login attempt and the next successful login for a given user [12]. They also performed some basic data cleaning, including removing duplicate log entries and malformed logs. Note that they used an individual log row as the unit for analysis, and did not use aggregation of log entries.

Our next and larger body of work uses authentication logs to create metrics and derivatives to directly identify insider threats and profile groups of users with similar behaviors. Recently, Sonneveld et al. published a study examining the non-intrusive security relevant information available to an SOC [13]. Through examining which resources users accessed, and when they were accessing them, they were able to identify each "ITAdmin" user. Using similar measures, they clustered users and tested deviation from cluster baselines as a potential indicator of insider threats. Carnegie Mellon's synthetic "Insider Threat" data set was used to test their methodology. Using this data set, they correctly detected 80% of insider threats in the ITAdmin group [6]. Intuitively, having high cluster consistency is key to getting a consistent measure for deviation; however, when they applied the clustering methodology to real-world data, consistency was cut in half. They attribute this partially to the much higher granularity of the real world data compared to the synthetic data. For other work clustering users, see Garchery and Freeman [5] [4].

Third, there is similar research that focuses on indicators of compromise or impersonation rather than insider threats, again using authentication logs to derive relevant metrics and measures. Liu et al. [7] created a behavior-based model to detect compromise using only two features: consecutive failures and login time of day. Their low computation-cost probabilistic model showed a good true positive to false positive trade off with high accuracy and low false positive ratio. They used a real-world private dataset of 4 million logs, and state that it contains no authentication compromises. This paper is of particular interest to us due to the unique way they construct derivative authentication "events" as their unit of analysis, rather than using individual log entries as atomic units. The authors aggregated raw log rows into series of 0-n failures prior to a success; series that don't result in success are discarded. The maximum gap between a failed log row and the following success is not stated. The resulting "events" do not include failures, and some "events" may span time periods longer than the user's interaction. See Bian et al [2] for similar work using to identify lateral movement.

Finally, we note the work of Alahmadi [1], who surveyed SOC practitioners investigating analysts perspectives on security alerts. They report an excessive number of alerts experienced across organizations, which contributes to analyst fatigue and human error. This is exacerbated by the low interpretability of the alerts being generated. These findings, in combination with Zhao et al. [14] who found that log data was used in over 30% of incident diagnoses, with indicators that this portion would be larger if the logs had greater interpretability, suggest the potential benefit that could accompany more interpretable logs and alerts.

III. Methodology for Constructing Authentication Events

Using data obtained through the University of Tulsa IT department, and approved for analysis by the Institutional Review Board (IRB), we developed a process to capture user authentication "events" from raw authentication logs. We define an event as:

> The occurrences reflected in log data that are directly experienced by a user, beginning when an authentication to a particular application is initiated, and terminated upon the eventual success, or abandonment of the authentication attempt.

By filtering sign-in logs to events directly experienced by the user, we can construct event-based metrics of usage and performance while reducing noise and increasing interpretability. In this section we provide an overview of the process to translate authentication logs to events, followed by a description of each step, and concluded with a description of the resulting events.

A. Process Overview

Before we dive into details of the process, we first give a high level example in Figure 1. The steps are:

1) **De-Identify**: These logs are first stripped of four direct identifiers which are replaced by the "Participant ID" attribute[1].
2) **Row Code**: Each row is assigned one of 46 "Row Codes" which captures both the overall success or failure result and detail about the action performed. This row code is the backbone of the encoding system, and will be explained in greater detail in III-C.
3) **Reduce**: Several helper attributes are added, such as "event number" to indicate which "event" a particular authentication entry is associated with. An attribute tracking if a password is entered is added by cross referencing an entry's "RequestID" with its entry(s) in the "authDetails" files. These attributes are used in combination with the row code to produce the "interactive" attribute. Duplicates and known or suspected malicious entries are removed.

[1]This step is only necessary in a research context where the users remain anonymous to the researcher.

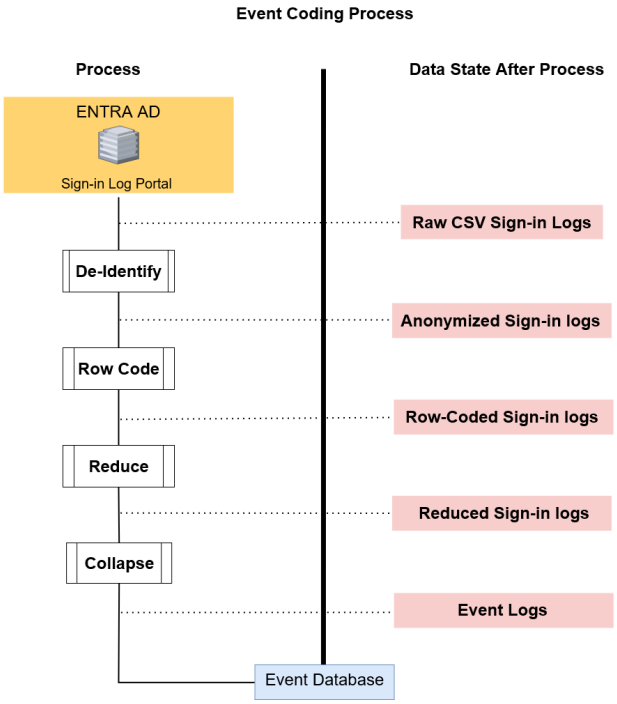

Fig. 1. Event coding overview diagram.

4) **Collapse**: Finally, we create a derivative data set by aggregating the key attributes from all rows for a given event, tracking the number and type of errors encountered, the form of MFA used, etc. This yields a smaller data set comprised of rows with 20 attributes, each row describing a complete authentication attempt to a particular application. The final set of attributes is easily adjusted based on the attributes available in the raw data.

B. Raw Log Data Description

The author's university utilizes Microsoft Entra AD for authentication. Data is first collected through the Entra AD portal, which presents the sign-in logs broken down into six categories. The first four categories are interactive and non-interactive sign-in logs and their corresponding interactive and non-interactive "auth details" files. The final two categories are called "Application logs" and "MSISignins"; these refer to authentications by service principals, and authentications by an Azure Managed Identity, respectively. Interactive logs are defined by Microsoft as those sign-ins where "a user provides an authentication factor, such as a password, a response through an MFA app, a biometric factor, or a QR code".

To investigate user experience, we ignore the application and managed identity logs, as they are not indicative of human interaction. Microsoft's labeling of interactive and non-interactive may be helpful in some cases; however, it does not strictly adhere to the definition of interactivity we use in this paper. For example, we want visibility into the errors that occur prior to or following presentation of an authentication factor. Thus, both files and their associated authentication details are downloaded. A single log entry contains 44 attributes

TABLE I
RAW AZURE AD SIGN-IN LOG ATTRIBUTES

Category	Attributes
Direct ID	User, User ID, Username, Sign-in Identifier
Device	Device ID, **Operating System**, Browser
Connection	IP Address, Location, Latency
Connection	IP (seen by resource)
Session Info	**Date (UTC)**, Application, Application ID
Session Info	Resource, Resource ID, Resource Tenant ID
Session Info	Home Tenant ID, Home Tenant Name, Request ID
Session Info	Correlation ID, Cross Tenant Access Type
Session Info	Incoming Token Type, Unique Token Identifier
Session Info	**Client app**, Client Credential Type
Session Info	Autonomous System Number, Token Issuer Type
Session Info	Incoming Token Type, Token Issuer Name
User Provenance	User Type, Compliant, Managed, Join Type
Authentication Info	Authentication Protocol, Conditional Access
Authentication Result Info	**Status**, **Sign-in Error Code**, Failure Reason
Result Info	**MFA Result**, **MFA Auth Method**, MFA Auth Detail

and describes a single system interaction. A small period of user interaction can generate several to dozens of log entries per minute, many of which may represent back-end processes that users don't directly experience within an authentication attempt. The process described below is implemented on Entra AD logs, but is designed to be generalize to other sources of authentication logs.

Table I summarizes the attributes, which we have split into 7 broad categories. *Direct ID* attributes identify the specific user, which are immediately removed and replaced with a unique user number. *Device* and *Connection* attributes detail the network connection and device characteristics. *Session Info* attributes comprise the bulk of the data, including the name and ID of the application and resource being used, token information, client application, and so on. Redundant attributes will be dropped in processing, such as alpha-numeric "ID" fields like "Resource ID"; as "Resource" is retained, which is the name of the Resource. The "Request ID" field is always retained, as it is the unique key linking a particular log item with other associated data in the Azure AD system. The *User Provenance* category includes information about a particular user's account, such as their user type (member or guest) and join type (Azure AD Registered, Azure AD Joined, Hybrid Azure AD Joined). The smallest category is *Authentication Info*. Relevant attributes include "Authentication Requirement", which indicates if the authentication requires single or multi-factor authentication, and "Conditional Access", which indicates any conditional access policies that were applied and the result. Finally, the *Result Info* category includes details about the authentication attempt and result.

The "Status" attribute has one of three values: Failure, Interrupted, and Success. Note that many "Failure" results are not caused by improper user action, and "Interrupted" results often do not tangibly disturb the user experience. The "Sign-in error code" attribute contains a numerical error code when an error is present, which is true for any entry that is not labeled "Sucesss". This error code is the key attribute used to assign

TABLE II
ROW CODES - SUCCESS (ALL)

Code	Item	Authentication
0	Token Success	Multi-Factor
1	App Password	Multi-Factor
2	Remembered Device	Multi-Factor
3	Registered Device	Multi-Factor
4	App Notification	Multi-Factor
5	SMS Verification	Multi-Factor
6	Phone Call	Multi-Factor
7	OATH	Multi-Factor
8	Token Authentication	Single-Factor
9	Password Authentication	Single-Factor
10	Password Authentication	Multi-Factor

TABLE III
ROW CODES - ERRORS (SAMPLE)

Code	Item	Type
9	Token Failure	Interrupt
10	Needs to Complete MFA	Interrupt
16	Device Code Expired	Interrupt
14	User has no Role in Application	Configuration
19	Error Issuing Token	Configuration
27	Failed to Complete MFA	User
30	Limit on MFA Calls	User
34	Blocked for Malicious IP	Hacker
45	Uncategorized Single Factor Error	Unknown
46	Uncategorized Multi-Factor Error	Unknown

row codes for non-pass rows. The "Failure reason" attribute contains a description of the error code result when an error is present, and detailed descriptions of errors and remediation are available from Microsoft on their website [8]. There are three MFA-related fields: "MFA result" provides a text description of the authentication result; "MFA auth method" contains the type of MFA used when applicable, and "MFA auth detail", which may contain a phone number associated with the MFA with the last two digits revealed. The last field is a Boolean "Flagged for review", which is only true when an admin flags a user account.

C. Row Coding

Adding a row code enables us to distill the 44 attributes included in raw log instances to a minimal expression. Thus, a set of 46 *row codes* were created to capture critical information about an authentication attempt's result. There are two broad results that a single entry can indicate: Pass (Success), or Fail, indicated by the attempt concluding in an entry marked "Failure" or "Interrupted" in the "Result" field of raw sign-in logs. A selection of row codes can be seen in Table II and Table III below.

Nine categories of logs were identified that indicate authentication has passed as seen in Table II. These 9 categories are variations of 3 basic results: Token Successes, Remembered Device Successes, and MFA Successes. Token Successes are split between single and multi-factor authentications, and all multi-factor authentications that are not token-related are either a primary form of MFA such as Text message, OATH, etc. or fulfilled through remembered device. Six row codes capture the various forms of MFA Successes, and two capture the remaining single factor successes.

The remaining row codes are used for entries that do not indicate an authentication pass, and we group these 36 row codes into 3 primary categories of errors: Interrupts, User Errors, and Configuration Errors.

Interrupts occur when the "Failure" (or Interruption) reported is not a true failure, it is a redirect or part of the intended authentication flow. In our user-centric paradigm, this means the user is not met with an error message, they do not experience a failure. One example is Row Code #9: Token Failure: it is not an error in the sense that the user or application had an issue; rather, it is an expected part of a token's life-cycle. When this Token Failure error occurs, a user has entered their password and asserts a token that would otherwise satisfy the second factor requirement, but that token is invalid for one of many reasons. The user experiences this as being directed to their MFA prompt screen after inputting their password. This is a typical use case and not experienced as failure or extra delay. "Interrupts" do not detract from typical user experience.

The key difference between user and configuration errors is the agency of the user to resolve the error. Any error that was either directly caused by the user, or is within the user's power to resolve, is considered a user error. 8 row codes are used for the user errors. For example, row code #27 indicates a user initiated a multi-factor sign in but never provided the second factor, and row code #26 indicates a user input an incorrect password.

An additional 8 row codes are used for configuration errors, which includes transient errors. Row Code #18 is a good example, wherein a user tries to authenticate to an application, but is denied because their account has no associated role in the application. The error message presented indicates that an administrator must give the user access, it can not be dynamically requested, making this an error outside direct control of the user. Finally, we have codes that capture behavior identified by Azure AD as malicious, and a catchall for uncategorized errors. We now describe the process of creating these row codes, beginning with non-pass entries. Three of the co-authors, two with high domain knowledge and one with low domain knowledge, independently inspected log samples encompassing each unique "Sign-in Error Code" present in the dataset. Co-authors labeled each error code with one of four categories: Interrupt, User Error, Configuration Error, or Hacking Error. Each error was considered alongside all available documentation and examples of the error appearing in the data. Krippendorff's alpha was 0.73 considering all three raters, and 0.86 for the two raters with high domain knowledge. Majority opinion was sufficient for all but one of 127 unique error codes labeled, and each labeling was reviewed and confirmed by the authors. Labeled errors were then grouped into row codes by similar themes within each category of error. These processes yielded the final set of 36 error groupings, which were then given integer representations

TABLE IV
EVENT LOG ATTRIBUTES.

Attribute	Category	Comments
Direct ID	User	Participant ID
Device	OS	String
Device	Browser	String
Connection	IP Address	Alpha-numeric
Session Info	Event#	Int
Session Info	Application	String
Session Info	Service	String
Session Info	ClientApp	String
Session Info	Start	DateTime
Session Info	End	DateTime
Auth Info	MFA Type	String
Auth Info	AuthReq.	Single/Multi-Factor
Result Info	Result	Success/Failure
Result Info	Detail	Result Details
Result Info	Password Entries	Int
Result Info	Elapsed	Elapsed Time in Seconds
Result Info	TA	Time Away in Minutes
Result Info	UEs	User Errors Count
Result Info	IEs	Int. Errors Count
Result Info	CEs	Config Errors Count
Result Info	Error Codes	Int List of Errors

beginning after the 10 "Pass" row codes. In total, 127 distinct sign-in error codes from the logs were mapped to 36 row codes. This is best explained using examples.

1) **Row Code 11**: There are 2 error codes that indicate MFA Completion is required. They redirect the user to use their second factor for the authentication, "Sign-in Error Code" 50074 and 50076.
2) **Row Code 18**: error codes 50105 and 50177 both describe a user who has not been granted specific access to an application, and is classified as a configuration error. This is distinct from a user who is dynamically requesting access to an application, which is classified as an interrupt, as it is an intended step in the authentication cycle, not the result of incorrect permissions or any failure.

D. Reduction

There are four steps taken to reduce the authentication logs after row coding. Here, we note that the focus of this paper and the authors' related research has been on measuring and characterizing legitimate use. As such, we discard known and suspected malicious authentication attempts when constructing events. First, we discard logs from non-standard user agents including POP and IMAP, and logs categorized as "Hacking Errors", such as those with row code #34: "Blocked for Malicious IP", as these attempts are unlikely to be from legitimate users interacting with our applications. Second, we discard logs from authentication attempts made to "API" resources, which are not indicative of interactive user authentication, as these are authentications performed by some user-side application to access a third party resource. Third, we discard duplicate logs, defined as logs with identical attributes occurring within one second of each other. Finally, we also discard any logs whose row codes are not labeled as interactive, which is a sub-attribute of our row codes. These reductions ensure we have non-redundant data that focuses on legitimate, interactive user behaviors and experiences.

E. Collapse into Events

Returning to our definition, we define an *event* as:

> The occurrences reflected in log data that are directly experienced by a user, beginning when an authentication to a particular application is initiated, and terminated upon the eventual success, or abandonment of the authentication attempt.

Each event captures the number of errors encountered before eventual success or failure, as well as the type of errors involved, time spent on an attempted authentication, and the type of authentication used. Since these characteristics are reflected in the row codes outlined above, tracking their occurrence in events is straightforward.

Events are constructed by aggregating rows with the same "Event Number". This number is created by first sorting entries by user and datetime, and setting a boolean "New Event" to TRUE if the gap between the current entry and prior entry exceeds 90 seconds. A cumulative sum is run on the "New Event" attribute to assign an event number to each log. In an enterprise environment without SSO implementation, a second condition is introduced: the successful completion of an authentication. In our SSO environment, once an authentication succeeds, any subsequent authentications to related sites will be non-interactive and fulfilled by the token presented by the user, resulting in no authentication interaction.

By defining events in this manner, we are flexible enough to accommodate situations where the user initiates multiple applications simultaneously. For example, a user might first be prompted for MFA on their desktop Outlook client. If that fails, a user could authenticate using a web-based interface instead. For our purposes, this is treated as a single event when occurring in close temporal proximity, which is effective for our enterprise environment in which there are many different applications which can be satisfied by completing authentication in any one service. The resulting "event" provides a clear indication of overall success, the application used, MFA Type, time spent, count and classification of errors, and provides the error codes associated with the errors to enable user and population metrics.

F. Event Examples

"Events" are comprised of the 21 attributes listed in Table IV. The first attributes tell us who authenticated, the system they used to do so, and total time elapsed. We also retain authentication information (MFA type and whether one or two factors were required). The final 9 attributes capture relevant details about the authentication experience by aggregating the observed row codes for log entries in the event. Note that a user can experience one or more errors, from misconfigurations to failed passwords or MFA prompts, before ultimately succeeding in the authentication. Such impediments are reflected in the other fields, such as the "Password Entries" attribute that tracks the number of times the user input their password during the authentication event. The "Elapsed" attribute is calculated by the difference between the first and

TABLE V
SAMPLE AUTHENTICATION EVENTS.

Ev #	Result	MFA Type	PW #	Time (s)	TTR (min)	OS	Application	UE #	CE #
3	Failure		0	2	4	Windows	Office 365	0	1
4	Success	App	2	16	NA	Windows 10	Azure Portal	1	0
6	Success	App	1	0	NA	Windows 10	Azure Portal	0	0
12	Failure		1	2	4	Windows	Office 365	0	1
13	Success	App	1	0	NA	Windows 10	Teams	0	0

last rows in a sequence that collapses into an event. Because there is no indicator in the raw sign-in logs when a Multi-factor prompt is initiated, this measure captures the extra time spent due to errors and interruptions in the authentication process. Time Away (TA) measures the gap in time between a failed authentication event and the next attempted login[2]. The final attributes tally the number of User, Interrupt, and Configuration Errors experienced during the authentication event.

Table V illustrates the "event" log with example events. Event #3 shows a simple failure with a single "Configuration Error" (CE). A "Time Away" of 4 minutes is listed, indicating that 4 minutes elapsed before the next successful authentication, event #4. Event #4: App-based MFA was used to successfully sign into the Azure Portal on a Windows device after a single "User Error" (UE), an invalid password entry. The authentication process took 16 seconds after initiation, significantly longer than that observed by [10], which is likely a consequence of the failed password entry. Event #6 offers another example of a simple success with no errors that takes 0 seconds after initiation to complete. This zero second time reflects the complete lack of friction in the event, as we do not know when the user started to input their password, use MFA, etc; we only know when the user hit ENTER or otherwise imitated the authentication. By breaking down authentication logs into discrete user-centric events, we can provided meaningful insight into the user experience and application health, as we demonstrate next.

IV. LEVERAGING EVENT DATA

For this section of the paper, we utilize a subset of collected data that centers around three semesters: Spring and Fall of 2022 and Spring of 2023. These slices include one week prior to the first day of class and end one week after the semester concludes; January 8th through May 17th for the spring semesters, and August 13th to December 19th for the Fall. After filtering for users that had at least one successful authentication, we are left with 1.7m events across 7,419 users, an average of 77 authentication events per user, per semester.

A. Basic Outcome Measures

The examples discussed in this section demonstrate the utility of user-focused event aggregates and their derivatives. A proactive SOC may directly utilize some of these capabilities beyond the standard authentication log use cases of alert diagnosis and incident response. For example, the detection of lapsed applications discussed in section IV-C could be used to reduce threat surfaces by retiring unused applications. As we consider the utility of an event-based approach to authentication logs in a SOC, we begin by examining the basic unit of analysis, the event, before moving on to derivative measures. As we see in Table V, each event reports success or failure, the time elapsed, the form of MFA used, types of errors encountered, and application being authenticated to. The most straightforward measure then, is failure rate, the complement of success rate.

An intuitive way to examine failure rates is by error content: does error type impact the user experience differently? We anticipate that errors caused by users are both more common and more easily resolved; passwords can be re-entered, MFA can be properly completed, etc. We find that over 80% of users who encounter a configuration error will never succeed when they experience a configuration error, and 94% of events containing a configuration error end in failure. Conversely, we find that only 7% of users who encounter user errors will never succeed when they experience a user error, and only 56% of events with user errors conclude in failure. While configuration errors are clearly more difficult to resolve, they are also less common. 93% of users experience user errors, while only 27% of users experience configuration errors. This confirms our expectation that user errors are both more common and more easily resolved.

Examining the cumulative distribution function (CDF) plots in Figure 2, the majority of our 7,305 valid users experience a very low failure rate. Mean failure rate is 8%, with the 10% worst users failing over 20% of authentications, and the 10% best users fail only 0.4%. The failure rate increases substantially for our worst users when we examine those who ever experience configuration errors, plotted here in red. The 10% best users fail only 1.5% of authentication events, whereas the 10% worst fail over 30% of authentications. The bottom 5% fail an astounding 47% of authentication attempts.

We can take away a few lessons from these distributions for utilizing event data in a SOC. First, configuration errors may be worth investigating, as they reliably trigger failures through no fault of the user. Second, relatively few users fail frequently, and it may be beneficial to target efforts at assisting these struggling users.

B. Identification of Locked-Out Users

Creating derivative metrics lends greater utility, such as the ability to identify locked-out users. An alert prompted by a lockout metric might trigger automated assistance, which in turn could forestall help tickets and issue early alerts for developer issues that cause service interruptions.

To construct this measure, we first add helper variables to our event dataset: we add a "consecutive failures" and "hours away" attribute to each event. Next, we set a variable "lockout" to true when consecutive failures is greater than one and time away exceeds twelve hours. Each week is summarized by the longest lockout experienced for each user.

[2]TA is similar to "recovery time" reported by [12], which captures the time between a failure and the next success.

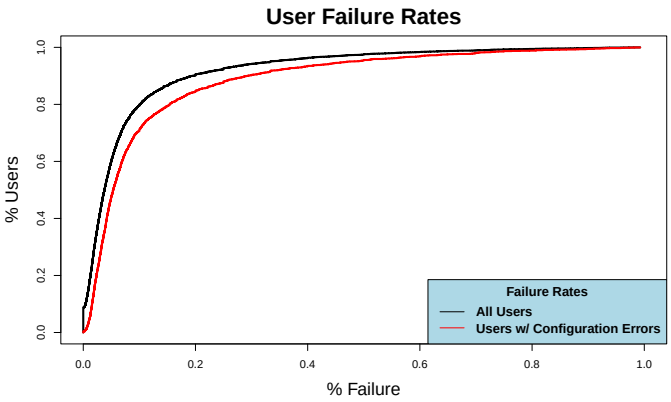

Fig. 2. Failure rate per user, considering errors.

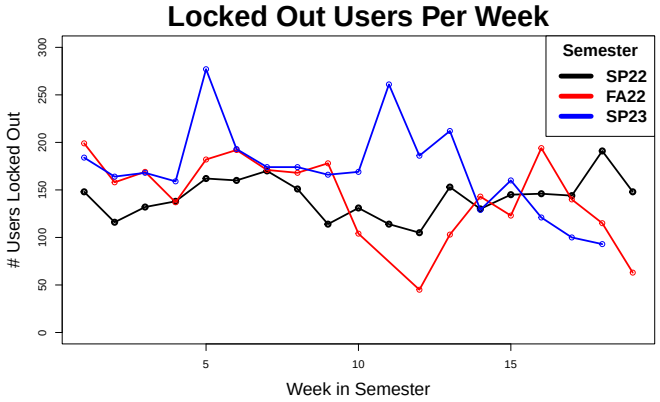

Fig. 3. Number of Locked out users per week in each semester.

Figure 3 shows the number of users locked out for each week of the semesters. The average number of users locked out for more than 12 hours, each week in the semester, was approximately 2.5% (152 of 6017) of the total. If we filter this for lockouts over 24 hours in duration, it shifts to 105 users per week, or 1.7% of our users.

Next we plot another set of CDFs, this time examining the duration of lockouts. As configuration errors affect failure rates more than user errors, we plot Figure 4 with a series of mixed errors in black, and a series with only configuration errors in red. Across three semester, we observe 8350 lockouts for 2656 unique users, which is 36% of our total user base. We note that nearly 93% of lockouts were associated with both user and configuration errors, and the mean ratio of CEs to UEs for those lockouts was 3.7. Over 6% of lockouts only had configuration errors, and less than 1% only had user errors. Lockouts commonly persist beyond 12 hours, with a median lockout duration of 43 hours, and the 90th percentile being locked out for over 193 hours, or 8 days. Lockout times begin to diverge based on error composition after the 24 hour mark and are longer when caused by configuration errors.

Lockouts happen often enough to benefit from proactive investigation and resolution, but they are uncommon enough to not overwhelm analysts. Moreover, since lockouts can persist for a long time, steps to eliminate them sooner would bring substantial value.

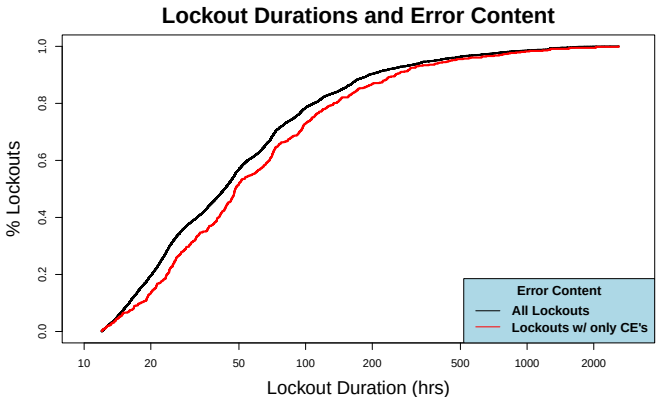

Fig. 4. Cumulative Distribution Function of Lockout Duration

C. Identification of Lapsed and Struggling Applications

Maintaining the security and performance of enterprise applications is a key function of a SOC. Applications that are unused and/or not associated with any successful authentications present a security risk; these applications are more likely to lapse into unsafe states, and misuse may be harder to detect. In our data, we observe 689 unique applications across three semesters, 348 of which never show a successful authentication. In our organization, over 50% of applications can be easily identified and classified as lapsed, and may be decommissioned to increase security.[3] These lapsed applications may otherwise persist for long periods of time, as we observe in the bar chart 6, which shows the number of valid and invalid applications per semester.

The next utility is early identification of struggling applications. Using the most recent semester, SP23, we first filter out the lapsed applications with no record of successful authentications. This results in a median success rate of 95% percent, closely matching our median user success rate for that semester of 94%. The mean success rate per application is somewhat lower, at 76%, indicating that some of our highly used applications have lower success rates. Examining the 20 most used applications, which in our data incur an average of 140 unique users per week, we plot the per application success rate over time to observe struggling applications. We define a struggling application as an application experiencing a success rate 50% below its mean success rate across the semester. In Figure 5 we report the lagging top 20 applications per week in the SP22 semester.

As one might expect, the top applications usually perform well, but it is not uncommon for one or a few to be lag-

[3]We do not currently posses a master list of applications for our organization, and can only detect applications with at least one authentication attempt made. In a SOC setting, this is easily remedied to be exhaustive and complete.

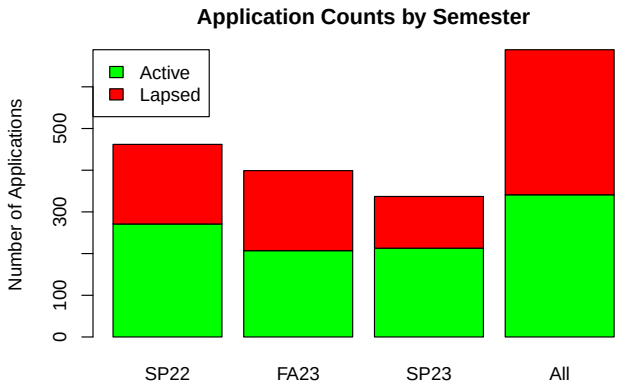

Fig. 5. Lapsed and Active Applications per Period

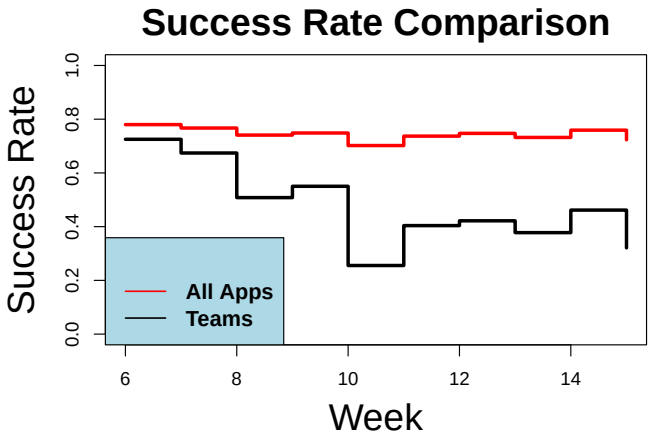

Fig. 7. Success Rate of Microsoft Teams vs All Applications

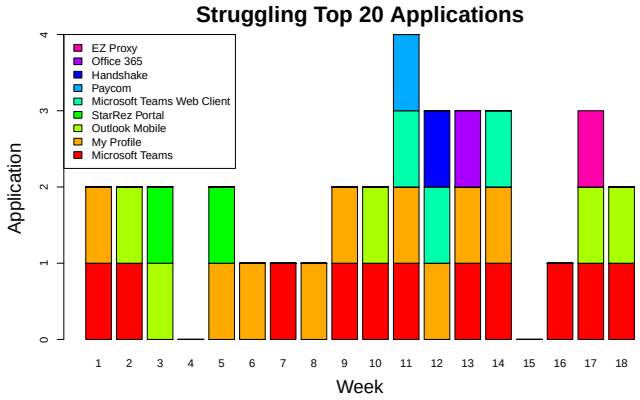

Fig. 6. Lagging Top Applications per Week

ging. Taking a specific application as an example, Microsoft Teams had a fairly low mean success rate of 53% in the SP23 semester, compared to 78% in the SP22 semester. Our "Lagging" metric flags a per-day success rate of under 10% on the last day of week 9, pointing to acute issues with the application. The graph in Figure 7 shows the downward trend of weekly success rate and its impact on the success rate across all applications in the following weeks. Early identification of such issues is key in reducing the impact of lagging applications on an organization.

V. Concluding Remarks

In this paper we described a process to distill raw authentication logs into more meaningful events, then applied that methodology to real-world data. The process utilized was designed to incorporate a level of domain knowledge to improve the utility of raw logs, but be broad enough to generalize to other sources of authentication logs. These examples are simple demonstrations of the type of utility the events provide, such as identifying struggling users and lapsed applications.

We contend that the event view developed for the analysis has the potential to improve SOC analysts' performance by providing a human readable summary of a user's experience that collapses numerous otherwise difficult to read log entries. This enables cybersecurity teams to quickly assess the state of a user's authentication and note changes in usage and performance patterns when investigating alerts. Finally, this new unit of analysis allows for the creation of event-based metrics that can better capture subtleties of authentication usage and performance. Future work is planned to develop and deploy an event-based dashboard in the university SOC. This will help to evaluate the measures and incorporate feedback from real-world usage.

This paper's primary goal was to describe and demonstrate a methodology for constructing user-focused authentication event logs. We attempt to filter out entries that do not reflect user interaction, but some events are inevitably missed. There is ongoing work by co-authors to utilize this event data in a diary study tracking users' authentication experiences. This should provide a valuable opportunity to validate the approach and examine if the events as constructed match the users' perceived experience.

In future research, we could apply event-based authentication logs to user clustering in a system that detects malicious activity. It is possible that our event-based log method would remove noise that may have contributed the lack of cluster consistency found by [13] in their 2023 study, and "tune" our tools to the input we're most interested in. Our approach also introduced derivative measures that embed a baseline of domain knowledge, such as distinctions between user errors and configuration errors, which can help differentiate two behaviors or experiences that might otherwise appear similar.

Acknowledgments

The authors thank Sal Aurigemma and Bradley Brummel for their feedback and acknowledge support from Tulsa Innovation Labs via the Cyber Fellows Initiative.

References

[1] B. A. Alahmadi, L. Axon, and I. Martinovic, "99% false positives: A qualitative study of SOC analysts' perspectives on

security alarms," in *31st USENIX Security Symposium (USENIX Security 22)*. Boston, MA: USENIX Association, Aug. 2022, pp. 2783–2800. [Online]. Available: https://www.usenix.org/conference/usenixsecurity22/presentation/alahmadi

[2] H. Bian, T. Bai, M. A. Salahuddin, N. Limam, A. A. Daya, and R. Boutaba, "Uncovering Lateral Movement Using Authentication Logs," *IEEE Transactions on Network and Service Management*, vol. 18, no. 1, pp. 1049–1063, Mar. 2021, conference Name: IEEE Transactions on Network and Service Management.

[3] J. Colnago, S. Devlin, M. Oates, C. Swoopes, L. Bauer, L. Cranor, and N. Christin, "It's not actually that horrible: Exploring adoption of two-factor authentication at a university," 04 2018, pp. 1–11.

[4] D. Freeman, S. Jain, M. Duermuth, B. Biggio, and G. Giacinto, "Who Are You? A Statistical Approach to Measuring User Authenticity," in *Proceedings 2016 Network and Distributed System Security Symposium*. San Diego, CA: Internet Society, 2016. [Online]. Available: https://www.ndss-symposium.org/wp-content/uploads/2017/09/who-are-you-statistical-approach-measuring-user-authenticity.pdf

[5] M. Garchery and M. Granitzer, "Identifying and Clustering Users for Unsupervised Intrusion Detection in Corporate Audit Sessions," in *2019 IEEE International Conference on Cognitive Computing (ICCC)*. Milan, Italy: IEEE, Jul. 2019, pp. 19–27. [Online]. Available: https://ieeexplore.ieee.org/document/8816990/

[6] B. Lindauer, "Insider Threat Test Dataset," 9 2020. [Online]. Available: {{https://kilthub.cmu.edu/articles/dataset/Insider_Threat_Test_Dataset/12841247}}

[7] M. Liu, V. Sachidananda, H. Peng, R. Patil, S. Muneeswaran, and M. Gurusamy, "Log-off: A novel behavior based authentication compromise detection approach," in *2022 19th Annual International Conference on Privacy, Security Trust (PST)*, 2022, pp. 1–10.

[8] Microsoft. (2024) Error documentation. [Online]. Available: https://login.microsoftonline.com/error

[9] G. Pannell and H. Ashman, "Anomaly Detection over User Profiles for Intrusion Detection," *Proceedings of the 8th Australian Information Security Mangement Conference*, vol. Edith Cowan University, p. 30th November 2010, 2010, medium: PDF Publisher: Security Research Institute (SRI), Edith Cowan University. [Online]. Available: http://ro.ecu.edu.au/ism/94

[10] K. Reese, "Evaluating the usability of two-factor authentication," 2018.

[11] K. Reese, T. Smith, J. Dutson, J. Armknecht, J. Cameron, and K. Seamons, "A usability study of five two-factor authentication methods," in *Proceedings of the Fifteenth USENIX Conference on Usable Privacy and Security*, ser. SOUPS'19. USA: USENIX Association, 2019, p. 357–370.

[12] J. Reynolds, N. Samarin, J. D. Barnes, T. Judd, J. Mason, M. Bailey, and S. Egelman, "Empirical measurement of systemic 2fa usability," in *USENIX Security Symposium*, 2020.

[13] J. J. Sonneveld, "Profiling users by access behaviour using data available to a security operations center," Jan. 2023, publisher: University of Twente. [Online]. Available: https://essay.utwente.nl/94221/

[14] N. Zhao, H. Wang, Z. Li, X. Peng, G. Wang, Z. Pan, Y. Wu, Z. Feng, X. Wen, W. Zhang, K. Sui, and D. Pei, "An empirical investigation of practical log anomaly detection for online service systems," *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pp. 1404–1415, Aug. 2021, conference Name: ESEC/FSE '21: 29th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering ISBN: 9781450385626 Place: Athens Greece Publisher: ACM. [Online]. Available: https://dl.acm.org/doi/10.1145/3468264.3473933