Account Lockouts: Characterizing and Preventing Account Denial-of-Service Attacks

Yu Liu[1], Matthew R. Squires[1], Curtis R. Taylor[1,2],
Robert J. Walls[1], and Craig A. Shue[1]
{ylu25, mrsquires, crtaylor, rjwalls, cshue}@wpi.edu

[1] Worcester Polytechnic Institute, Worcester, MA 10609
[2] Oak Ridge National Laboratory, Oak Ridge, TN 37830

Abstract. To stymie password guessing attacks, many systems lock an account after a given number of failed authentication attempts, preventing access even if proper credentials are later provided. Combined with the proliferation of single sign-on providers, adversaries can use relatively few resources to launch large-scale application-level denial-of-service attacks against targeted user accounts by deliberately providing incorrect credentials across multiple authentication attempts.

In this paper, we measure the extent to which this vulnerability exists in production systems. We focus on Microsoft services, which are used in many organizations, to identify exposed authentication points. We measure 2,066 organizations and found between 58% and 77% of organizations expose authentication portals that are vulnerable to account lockout attacks. Such attacks can be completely successful with only 13 KBytes/second of attack traffic. We then propose and evaluate a set of lockout bypass mechanisms for legitimate users. Our performance and security evaluation shows these solutions are effective while introducing little overhead to the network and systems.

Keywords: Account Lockout · Denial-of-Service (DoS) Attack · Single Sign-On · Middleboxes · Measurement

1 Introduction

In an attempt to gain unauthorized access to a system, attackers may try to guess the credentials associated with a legitimate user's account. These attackers may vary in sophistication, from brute-forcing passwords on default usernames to using a list of known usernames at an organization and lists of most commonly used passwords. Prior analysis of password data sets has shown that end-users often select weak passwords that are vulnerable to such attacks [37]. Further, many organizations consider usability and memorability to be key goals in username generation. As a result, usernames are often generated that match email addresses and use parts of a user's real name [38].

Given this threat, many systems implement an account lockout mechanism in which all authentication attempts are denied after a certain number of failed attempts in a predetermined time window. NIST, which sets standards for US government systems, recommends an attempt threshold of 100 attempts or less with a lockout period of between 30 seconds and 60 minutes [2]. The SANS institute recommends a threshold of five attempts with a 30 minute lockout period [30]. To be PCI compliant, which is required for organizations handling consumer payment information, accounts must be locked out for 30 minutes after six failed attempts [26].

This account lockout mechanism can be used by attackers to create a denial-of-service (DoS) attack that prevents legitimate users from gaining access to their accounts [22]. Such an attack is easy to launch: an attacker can issue authentication attempts at a rate that would keep an account perpetually locked. With the aforementioned thresholds, such an attack would consume minimal attacker bandwidth and computational resources. Even if simple IP address blocking is used for repeated failed attempts, an attacker could use a network of compromised machines to distribute the attempts.

With the deployment of single-sign-on (SSO) services, account lockouts can transform from a simple nuisance to a crippling attack. Recent work has explored web-based SSO systems and the relationships between identity providers that authenticate users and other websites, called relying parties, that use those identity providers to authenticate their own users [10]. In one example, a single identity provider was used by 42,232 relying parties. Further, recent reports [28] estimate that Active Directory—Microsoft's prominent SSO identity provider—is used in more than 90% of companies in the Fortune 1000. With an account lockout attack on a single identity provider, a targeted user could be denied access to thousands of other services.

In this paper, we ask two key research questions: *To what extent are organizations vulnerable to account lockout attacks? What countermeasures can be effectively deployed to address these attacks in a way that supports even legacy systems and devices?* Given its widespread deployment and integral nature at organizations, we focus our investigation on Microsoft's Active Directory service. In doing so, we make the following contributions:

1. **Vulnerability Measurements:** We examine 2,066 organizations, including Fortune 1000 companies and universities, to determine the extent to which attackers can systematically identify vulnerable authentication portals and lock accounts. We find that roughly 58% of the universities and roughly 77% of the companies examined expose a vulnerable authentication portal. Lockouts targeting these portals can potentially deny users access to thousands of applications [23].
2. **A Suite of Proposed Countermeasures:** Rather than relying on changes to Active Directory, we propose countermeasures that can be deployed immediately on legacy architectures. The suite of options, based on the concept of distinct authentication pools, includes mechanisms that work across devices with end-user involvement to completely transparent options, such as those using web browsers or modified home routers.

3. **Evaluation of the Countermeasures:** We evaluate the security effectiveness and performance of the proposed countermeasures. We find that each has clear availability advantages while introducing minimal performance costs. Notably, we find that existing authentication mechanisms—such as multi-factor authentication—are insufficient to stop account lockout attacks because the root problem lies with the lockout policy, not the mechanism.

2 Background and Related Work

The combination of a username and password is a ubiquitous method of user authentication. Attackers try to obtain such sensitive information to infiltrate computer systems. The sophistication of theses attempts vary. The most basic attack, a brute force attack, exhaustively enumerates all possible character combinations until a valid sequence allows access. The success rate of brute force attacks is dependent upon the underlying strength of user passwords [39].

Other approaches are more sophisticated and use information about end-user behavior to increase success rates [12]. Dictionary attacks, for example, form the password guesses by using a large database of popular passwords or words in a targeted language's dictionary. Prior research has found that many end-users select passwords that could easily be discovered by a dictionary attack [37]. Further, discovered passwords from a compromised service may be used to guess passwords for the same user at other sites due to password reuse [14].

To combat password guessing attacks, standard bodies recommend account lockout thresholds [2, 26, 30]. After a specified number of failed login attempts, the account lockout approach denies access to a given account even if valid credentials are provided [8]. This simple mechanism makes brute force password guessing infeasible and limits the rate at which attackers can make attempts using dictionaries. Unfortunately, account lockouts provide a natural avenue for denial-of-service attacks: an adversary can simply make numerous failed authentication attempts for a given username, causing the account to lock, and thereby preventing the legitimate account user from authenticating, as shown in Figure 1.

Other techniques attempt to limit malicious authentication attempts without using a lockout. A common approach is a form of automated Turing test before each login attempt that will purportedly distinguish a human from an automated adversary. A prominent approach is the CAPTCHA [27], which requires a user to decode an image or audio signal in a way that is challenging for computers to do. Such approaches may help deter dictionary attacks, but they do impose usability costs upon users [3]. Unfortunately, with innovations in machine learning, some previously-effective CAPTCHAs may be defeated automatically [36]. Further, hardware or legacy systems may be unable to support CAPTCHAs.

Aura et al. [4] propose the use of client-side puzzles to defend against denial-of-service (DoS) attacks by slowing the attack rate. Each time the client makes a request to the server, it is asked to solve a cryptographic puzzle provided by the server. These puzzles must require significant client effort to solve and are unpredictable. The verification of the result should be inexpensive. Dean et al. [6]

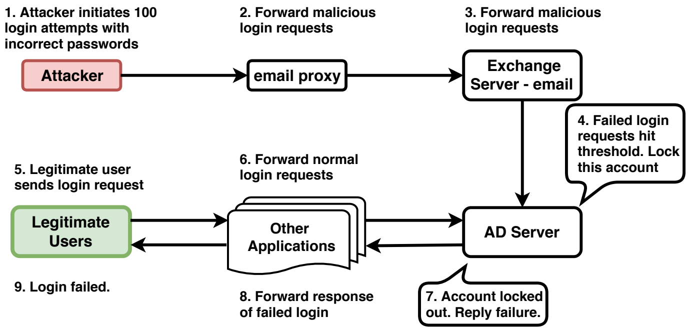
1. Attacker initiates 100 login attempts with incorrect passwords
Attacker
2. Forward malicious login requests
email proxy
3. Forward malicious login requests
Exchange Server - email
4. Failed login requests hit threshold. Lock this account
5. Legitimate user sends login request
6. Forward normal login requests
Legitimate Users
Other Applications
AD Server
9. Login failed.
8. Forward response of failed login
7. Account locked out. Reply failure.

Fig. 1. In an account lockout attack, the attacker selects a username and tries to authenticate with invalid passwords. Each failed attempt causes the server to increment the failed attempt counter for that specific account. When the legitimate user attempts to authenticate, the account may already be locked.

incorporate cryptographic puzzles into the TLS protocol to protect servers from DoS attacks. Koh et al. [15] evaluated a high performance puzzle algorithm.

Unfortunately, puzzle-based defenses may not be compatible with some existing systems and applications. For example, the use of a CAPTCHA may not be feasible when logging into a legacy video conferencing system. Some prior versions of mail software, such as Outlook 2010, not support tools like CAPTCHAs when authenticating. Similar limitations may occur for Skype for Business and applications without a browser-based interface.

2.1 Other Application-Layer Availability Attacks

Most denial-of-service (DoS) availability attacks target a bottleneck resource and overwhelm it to prevent legitimate user access. Network-based flooding attacks, for example, attempt to saturate the bottleneck bandwidth between the Internet and a targeted victim. Application-layer DoS attacks exploit a bottleneck in the host software to deny access. Moore et al. [24] describe an application-layer threat between an HTTP sever and a backend database resource. The account lockout attack is a variant of an application-level availability attack [22].

Source IP address filtering tries to mitigate a DoS attack by blocking the machines originating the attack. Unfortunately, modern attackers have botnets with millions of machines. By strategically cycling the attack machines, IP blacklisting techniques can be rendered useless since an attacker would have a large supply of previously-unseen machines that could trigger a lockout.

2.2 Active Directory (AD)

Active Directory (AD) is a service from Microsoft for managing user accounts and system resources belonging to an organization. It groups users, workstations, servers, and policies and organizes them into hierarchies that facilitate management. This service allows user management to be logically centralized by an organization in a set of domain controllers. Application servers may authenticate users via these domain controllers rather than managing accounts and

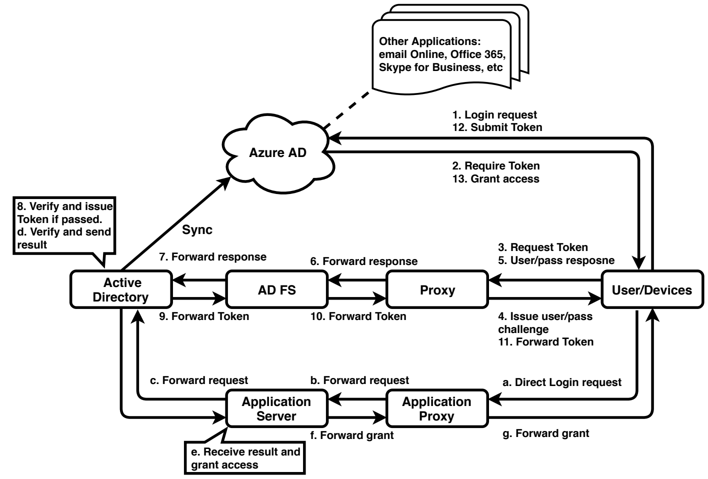

Fig. 2. This diagram depicts the components and interactions between an on-site AD domain controller, an Azure AD domain controller, and the ADFS connector.

passwords locally, as shown in Figure 2. For example, Microsoft's email server, Exchange, uses an AD server for authentication.

Organizations may host their AD domain controllers on-site, host them in the cloud through Microsoft's Azure AD service, or use a hybrid of both options. The Azure AD service is essential for Microsoft-hosted online services like Office 365 and Skype for Business Online. Those Azure-based services communicate directly with the Azure AD domain controllers rather than using the on-premise servers. In hybrid deployments, on-site AD domain controllers may configure a unidirectional synchronization channel with the Azure AD servers. For the purposes of account lockouts, an account locked by an on-site domain controller will result in a lock in all domain controllers and, depending upon the configured settings, may propagate to Azure domain controllers. In contrast, an account locked by an Azure domain controller will not propagate to on-site domain controllers.

Microsoft also provides an Active Directory Federation Services (ADFS) interface for applications to interact with Active Directory when they cannot use the integrated Windows authentication service. ADFS has its own account lockout mechanism, but that lockout only affects ADFS services.

For an attacker to maximize the impact of an account lockout, the best option would be to target a service that authenticates to an on-site domain controller, if such a domain controller exists, in addition to targeting an Azure AD domain controller. An account lockout in ADFS or in an Azure AD domain controller may result in a lockout that affects only a subset of the organization's services. However, in some attack scenarios, a subset may be acceptable to an attacker if it includes a critical service the attacker wishes to make unavailable.

2.3 Middleboxes for Security

Middleboxes, such as firewalls, intrusion detection systems, and proxies, have regularly been used for security purposes in the enterprise. Recent techniques have leveraged the cloud for enterprise security [31].

Other work has extended middlebox techniques to residential networks, including for whole-home proxies [35], validating TLS connections [34], and verifying IoT device communication [16]. As demonstrated by Taylor et al. [33], these residential middleboxes are feasible in countries like the United States since most residential users are within 50 milliseconds of a public cloud data center, causing middleboxes to only incur minor latency costs.

Our work shows how middleboxes can address account lockouts on an enterprise network in a backwards-compatible manner. We further show that middleboxes at the home (e.g. via a modified home router) can further enable robust account lockout protections.

3 System Overview

Active Directory is inherently flexible and scalable, which can lead to deployments that vary greatly in terms of complexity and redundancy. In the simplest case, an Active Directory setup involves a primary domain controller, one or more dependent application servers, and a set of client machines that wish to use the application server. Organizations may deploy other infrastructure, such as secondary domain controllers, proxy servers, and middleboxes, to support legacy systems or to achieve resiliency or security goals. Such infrastructure has little impact on the account lockout threat and we omit it for simplicity.

3.1 Assumptions and Threat Model

In the context of this work, the goal of the adversary is to deny a legitimate user access to services and resources through an account lockout attack. These adversaries may perform reconnaissance on an organization ahead of an attack to obtain email addresses, usernames, or to locate public-facing authentication portals. With the availability of botnets, an adversary may have significant computational and network resources. These resources afford the attacker significant flexibility in devising her attack strategy. For example, the attacker may send a high-volume of authentication requests from geographically-diverse machines and rapidly switch between IP addresses to avoid IP-based blacklisting.

This work does not consider attempts to compromise the Active Directory server, its dependent servers, or other hardware such as the organization's switches and routers. If these servers fall under the control of the adversary, it would be impossible for an organization to guarantee the accuracy of a user's identity or the availability of authentication services. Similarly, we assume an adversary lacks valid user credentials.

The defender's goal is to provide legitimate users with the ability to authenticate even under an ongoing lockout attack. For our proposed countermeasures, we assume that the organization's IT staff can insert one or more middleboxes into an organization's infrastructure, but they cannot modify the Active Directory server or the services that authenticate against Active Directory.

4 Characterizing the Account Lockout Problem

In this section, we explore the following research questions:

1. Can attackers feasibly exploit public authentication portals to launch account lockouts? With the help of a cooperating organization, we use only public data to effect an account lockout on a test account in a production environment using Microsoft's Active Directory service. Since that organization follows industry best practices and standards, this experiment is likely representative of many other organizations.
2. Can attackers automatically discover organizations' authentication portals for lockout attacks? Using an Internet measurement study, we show that authentication portals can be easily discovered by attackers.

4.1 Case Study: Identifying the Attack Surface in Production

We contacted a multinational organization with over 5,000 employees that uses Active Directory extensively and gained their approval to assess the impact of account lockouts across their environment. This organization used Active Directory for authentication for the vast majority of their IT services. From this case study, we created an Internet measurement strategy to characterize the risks at other organizations to determine the broader applicability of our findings.

Our partnering organization made an Active Directory administrator available to provide feedback on our tests, but the organization required anonymity as part of their participation. The organization created a test account for our use which was modeled after a standard employee account at the organization. The organization set a secure password on the account and ensured it was not shared with the authors performing the authentication attempts.

In our testing, we independently gathered information that was available publicly without use of organizational insider knowledge. In our experiments, we found that the organization used `mail.[organization domain]` to forward to a themed Outlook Web App (OWA) portal, which is a Microsoft-provided interface for web-based email. Since the portal used IP addresses that were not associated with Microsoft, we determine that the OWA portal was not Azure-hosted and thus was not using an Azure AD server for authentication.

With many Office 365 services, Microsoft provides a centralized authentication portal that leverages the user's email address to determine the appropriate Azure AD server to use to process the authentication. That authentication page compares the host portion of the email address to its list of registered organization domains. Accordingly, we went to the Office 365 authentication page [21] and entered a randomly constructed username, followed by the '@' character, and then the organization's domain name. The website redirected to the organization's account authentication page, where we were prompted to enter a password. This interface appeared to be Azure-hosted, indicating that login attempts would be directed to an Azure AD server. Account lockouts generated on this service would likely only affect services authenticating against the Azure AD server while not affecting user access via the OWA page we previously discovered.

We then examined Skype for Business (SFB, formerly known as Microsoft Lync). Based on Microsoft's documentation, the `lyncdiscover.[organization domain]` host name is typically used for this service. We performed a `CNAME` query on that host and the response indicated that the organization was not using an Azure-hosted SFB service. We then performed a DNS `A` record query, which returned a valid IP address that is not associated with Microsoft's Azure data centers, which suggests that the organization uses an on-site SFB service.

4.2 Case Study: Testing Account Lockouts in Production

After identifying the attack surfaces of the measured organization, we began testing account lockouts. Microsoft's documentation for Windows Server 2012 [17] and 2016 [18] recommends an account lockout of 10 attempts with a lockout period of 15 minutes. For Azure's AD service, Microsoft's documentation indicates a threshold of 10 attempts with a 1 minute lockout period. The most generous lockout policy was suggested by NIST with up to 100 attempts and a lockout period as short as 30 seconds. Based on these thresholds, we created an attack that would try authenticating as our test account with randomly-generated passwords around 200 times per minute. This attack is relatively low bandwidth at only 13 KBytes/second, which poses little burden on the attacker or on the organization's infrastructure. However, under the most conservative guidance, the attack would keep the targeted account perpetually locked.

We first targeted our attack at the organization's OWA portal. Our organization contact confirmed that the attack caused the account to be locked at the organization, preventing the account from logging into the organization's resources for the duration of the attack. We discontinued the attack and the organization contact removed the account lockout.

We next performed an attack targeted at SFB. Using the fake account, we use a tool provided by an open source project on Github named `lyncsmash` [25]. It provides an option to discover the SFB servers and an option to launch an account lockout attack. We manually went to the URL found in the tool, entered the username supplied by the organization, and entered an inaccurate password 10 times. Our organization contact then confirmed that the account was locked. We note that the `lyncsmash` tool can automate these attempts.

In these tests, we used the same source IP address for each query. While a simple IP rate limit or blacklist would stop our attack, an actual attacker could easily perform the attempts using a botnet to ensure no IP address queried more than once. This would easily keep the account locked without an obvious defense. The measured organization's contact confirmed that the organization lacks a mechanism to combat such account lockouts.

4.3 Characterizing the Risk with Internet Measurements

While our partner organization was vulnerable to an account lockout attack, we now focus on determining the extent to which other organizations are likewise vulnerable. We begin by making non-invasive measurements of the public-facing infrastructure of a set of organizations. While we focus on Active Directory in this work, most organizations avoid directly exposing their AD servers to the public for security reasons. However, in many cases, these organizations expose their application servers to boost productivity. To allow employees to access their email outside the office, these organizations may expose Exchange email servers or website interfaces, such as the popular Outlook Web App (OWA) that Microsoft provides. Unified messaging services, like Microsoft Skype for Business (SFB), allow employees, customers, and partners to instant message, call, and join video conferences remotely. In some cases, the devices joining these calls may be mobile phones or dedicated video conferencing hardware.

Given the popularity of email and unified messaging, our measurement study focuses on determining the extent to which authentication portals for Microsoft-specific email and messaging servers are exposed publicly since we know such servers must use an AD server for authentication. We perform our measurements by using a list of domains associated with the Fortune 1000 companies [11] and with 1,066 universities [32]. We focus on these organizations because their domains can be easily obtained. Further, these larger organizations likely have need for centralized authentication services like Active Directory.

Using our list of domains, we perform a DNS `MX` record lookup on the provided domain to determine the identity of the organization's public SMTP server. The host names of the SMTP servers provide some insight into the underlying infrastructure. For example, host names ending with `.protection.outlook.com` are indicative of an organization using Microsoft's cloud-hosted email service. Since these organizations necessarily use Active Directory in Microsoft's Azure cloud, these servers can be used to initiate an account lockout for all Azure-hosted solutions at the organization. Other `MX` records may indicate that the mail server is located on-site at the organization or is hosted by another provider.

Our second measurement uses information related to email auto-discovery [19]. We issue `CNAME` queries for the host `autodiscover` associated with the organization's domain (e.g., `autodiscover.example.com`). In some cases, the `CNAME` result was `autodiscover.outlook.com`, indicating the mail services use Microsoft's Azure-hosted Exchange server. In the case when another host name was returned, the mail server was not Azure-hosted. We then issued a web request on port 80 or 443 to the host name returned in the `CNAME` record. In some cases, the server required valid credentials to proceed. In some cases, the credentials would be validated by an Active Directory server, enabling the account lockout attack. However, in other cases, the authentication credentials could be independent of a user account (e.g., a username and password shared across the organization for relatively weak protection).

The discovered mail server's default web page could reveal information about the infrastructure. In some cases, the servers presented a default or themed

Organizations	Exchange Email		Skype For Business		Extent Vulnerable
	On-site	Azure-hosted	On-site	Azure-hosted	
Fortune 1000	190	339	360	345	765 (76.5%)
Universities	126	416	124	395	616 (57.8%)

Table 1. Our measurement study results show the majority of each group uses Microsoft services and has at least one exposed authentication portal, enabling account lockout attacks. The final column shows unique organizations vulnerable, even if an organization has multiple exposed attack surfaces.

version of Microsoft's Outlook Web Application (OWA) page, which is commonly associated with an on-premises Exchange server. When web servers return 403 forbidden, it means there could be a portal which requires authentication. We simply append "/owa" or "/autodiscover" and we found half of them redirect to an OWA login page. In other cases, the web server returned pages containing the string "Microsoft Corporation" indicating this server runs Microsoft's software. These authentication portals provide an avenue for the account lockout attack.

Some domains did not use an auto-discovery service or did not provide an obvious account authentication page. For these domains, we issued an `A` record DNS query for the `mail` host name associated with the domain (e.g., `mail.example.com`), which follows the examples provided in Microsoft's documentation for configuring mail servers. We found that nearly half of organizations provide such a server for their employees to authenticate, though few of them used a default interface such as OWA or Microsoft's Azure-hosted email portal.

We next focused our measurements on the Skype for Business (SFB) service. Microsoft's SFB client automatically searches for an organization's servers using a mechanism similar to email auto-discovery. For all the Fortune 1000 and university domains, we perform a `CNAME` DNS query on the `lyncdiscover` host associated with the organization (e.g., `lyncdiscover.example.com`), which can reveal which organizations use SFB services. We also query for `dialin.example.com` and `meet.example.com`, which are other commonly used SFB host names. When organizations use Microsoft's Azure hosted systems, the `CNAME` query returns an answer associated with the `webdir.online.lync.com` host name. For all the non-Azure responses, we performed a `A` record DNS query to obtain the IP address of the on-site SFB service.

In Table 1, we show the result of the measurements. Roughly 77% of companies and 58% of universities had servers that would be affected by some form of account lockout attack. For organizations that use Azure-hosted services, an account lockout attack targeted at these servers would affect other services that consult the Azure Active Directory server, but they would not affect services that communicate with an on-site Active Directory server because the uni-directional Azure AD server connection with an on-site AD server does provide the capability to share this information. However, attacks against services that communicate to a non-Azure AD server would affect all services, since non-Azure AD servers propagate an account lockout organization-wide, including to the Azure

AD server. Accordingly, attackers looking for the biggest impact may target non-Azure AD servers when possible.

5 Discussion of Potential Countermeasures

Our measurements demonstrate that account lockout attacks can be crippling for an organization and that many large organizations are vulnerable to these attacks. However, there are a variety of mechanisms that may be effective at mitigating such attacks. Each method has strengths and limitations in terms of ease of deployment, legacy compatibility, visibility and impact on end-users. We discuss potential methods and implemented two of them, one which modifies a residential router and another that leverages user provided secret information, to show to what extent we can prevent account lockout attacks.

Countermeasure: Private Usernames. An account lockout attack requires knowledge of the target username. In practice, gaining this knowledge is often trivial. For example, Alice's username might be `alice` and her email address may be `alice@example.com`. Intuitively, if the username becomes harder to guess then lockout attacks become commensurately harder for the attacker to execute. Private usernames offer tangible benefits. The approach is backwards-compatible with all existing infrastructure, it avoids lockout attempts on the username, and incurs no additional computational overheads or infrastructure. However, the approach may sacrifice end-user convenience for this computational efficiency. In particular, end-users will now need to manage multiple identifiers and know when to enter their private username and when to use their public email address. Further, organizations may need to reconsider how access control systems and resource sharing will work when a username is intended to be kept private from an employee's coworkers. Finally, transitioning to a private username schema may be prohibitively disruptive for organizations that have a large number of users and legacy systems.

Countermeasure: Multi-Factor Authentication. Another countermeasure is to ask the user to provide additional secret information as part of a multi-factor authentication (MFA) scheme, such as biometrics, hardware tokens, or one-time pass codes that are transmitted via a smartphone application. MFA-based approaches are effective at distinguishing legitimate users from attackers, assuming the attacker has not compromised all the factors. Further, they are widely deployed so users are already familiar with the process and the usability cost is relatively low compared to the security benefits. Unfortunately, multi-factor schemes alone cannot solve the problem of account lockouts. Intuitively, this problem is not necessarily a limitation of multi-factor authentication but of the lockout policies themselves. In other words, most lockout polices only account for the *number* of failed attempts and not the *kind* of information used in the attempt. Consider Active Directory's multi-factor authentication interface; this workflow allows a username and password to be used in conjunction with a second factor verification via smartphone application or text message. Failed authentication attempts still lead to an account lockout as the second factor is only used if the provided username and password are valid. In short, the second factor does not influence the server's decision to lockout an account.

Countermeasure: Observed Characteristics. The above approaches rely on the user to provide private information as proof. An orthogonal approach is for the authenticating server to use historical information related to the user's behavior or observable connection characteristics. For example, Eriksson et al. [9] studied geographic detection based on IP addresses. The primary limitation of such approaches is they must be tuned carefully to balance between false negatives (allowing an attacker to authenticate) and false positives (preventing a legitimate user from authenticating).

5.1 Distinct Authentication Pools

The goal of this work is to incorporate and augment existing authentication approaches. Our proposed countermeasures are based on the following observations. First, existing authentication mechanisms fail to stop account lockout attacks because the problem lies with the lockout policy not the mechanism. Second, account lockout policies should base lockout decisions on the totality of information rather than a simple boolean log of attempts. Third, the proliferation of legacy systems means that organizations are more likely to adopt defenses (at least in the short term) that do not require changes to the end-user software or existing authentication servers.

We codify these observations into a proposed authentication scheme based on *distinct authentication pools*. This scheme is designed to leverage historical activity, network proximity, and secondary credentials to maintain separate authentication risk pools with their own lockout thresholds and failed authentication attempt counts, as shown in Figure 3. Each pool maintains a separate counter and threshold which can be configured to meet different security requirements. To make authentication pools immediately applicable to existing systems, we use security middleboxes to implement the key functionality without requiring changes to the services or Active Directory servers. Importantly, this scheme allows a user to authenticate even if there is an on-going lockout attack.

We also propose and implement two novel, and orthogonal, authentication mechanisms to serve as the basis for two of the authentication pools. The first is a token-based mechanism that transparently authenticates requests originating from a user's residential network. The second proposed mechanism leverages a user-supplied credential that effectively turns a public username into a private username. We discuss the design of both mechanisms below.

5.2 Protecting Requests from Residential Networks

Many existing web-based authentication systems leverage HTTP cookies to determine if a user is currently logged in or has logged in successfully in the past. The pool proxy server could validate cookie values and place users with valid

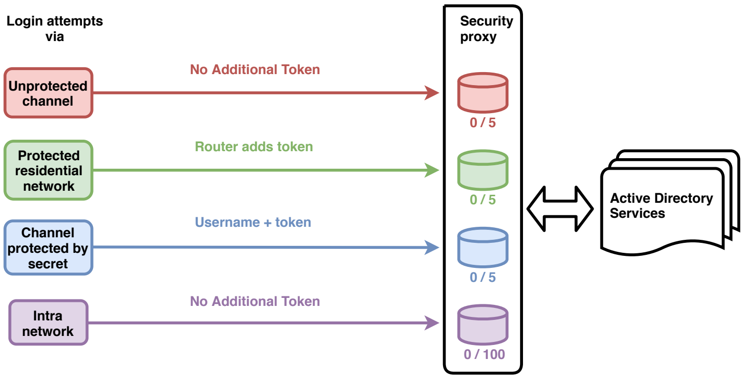

Fig. 3. With a security middlebox or proxy, an organization can create separate authentication thresholds and authentication attempt counts based on factors that may indicate the user's legitimacy. This diagram depicts four authentication pools based on the presence of tokens, manual authenticators, and on-site presence. The proxy can send commands to help account management in Active Directory, such as unlocking an account, restoring the original account lock state, and checking the account status.

cookies in a pool that is separate from users that do not present such cookies. Unfortunately, such an approach is limited to web-based authentication. Instead, we propose leveraging an approach from the TCP Fast Open (TFO) standard [5]. In essence, an authentication server orders the client or a middlebox to store a cookie, allowing that client or middlebox to prove it previously logged in successfully when trying to authenticate again in the future. Like the original TFO standard, the cookie we introduce would not be an authoritative authenticator and it would not be resistant to a man-in-the-middle attack. However, it does provide sufficient evidence to put a client in a separate authentication risk pool.

The authentication pool proxies can be implemented using the TLS "peek and splice" technique [29], in which the proxy is on the route to the protected application servers and has the private TLS keys associated with each server. This allows the proxy to decrypt the traffic, extract the username, and validate tokens and cookie values. If a token is present, the proxy server can then issue commands to the Active Directory server to determine whether the account is locked out, and if so, temporarily lift the lock. It can then re-encrypt the request with the tokens extracted and send it to the application server to process the authentication attempt. Once the authentication result is sent back to the proxy, it can re-lock the account, if it was previously locked.

In our implementation, to check if an account is locked and obtain the number of failed authentication attempts, the proxy speaks the LDAP protocol with the AD server. Using the `python-ldap` library [7], the security proxy checks the account status via the `ldap.search()` function on the `badPwdCount` attribute with administrator privileges. To unlock the account, the proxy server uses the `ldap.modify_s()` function and sets value of `lockoutTime` to 0. Finally, to restore the failed authentication attempts, the proxy server can use an invalid password to send the same number of prior authentication attempts.

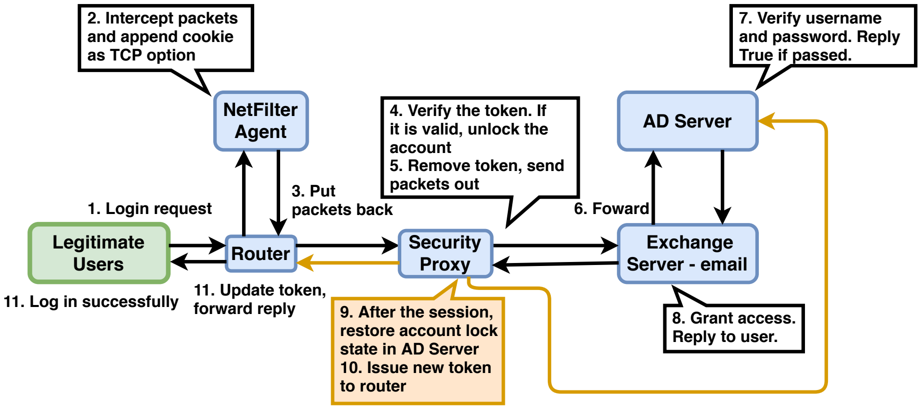
2. Intercept packets and append cookie as TCP option
NetFilter Agent
7. Verify username and password. Reply True if passed.
AD Server
4. Verify the token. If it is valid, unlock the account
5. Remove token, send packets out
3. Put packets back
1. Login request
6. Foward
Legitimate Users
Router
Security Proxy
Exchange Server - email
11. Log in successfully
11. Update token, forward reply
9. After the session, restore account lock state in AD Server
10. Issue new token to router
8. Grant access. Reply to user.

Fig. 4. With a residential middlebox, tokens can be automatically supplied and stored by examining packets to and from application servers.

For cookies using TCP options, the process can proceed in a fashion similar to TCP Fast Open (see Figure 4). Organizations can provide some employees with a modified router that will act as a middlebox that manages TCP cookies for the user. When the user accesses the organization's servers, the router checks to see if it has a cookie for the destination. If so, it adds the cookie as a TCP option with the request. The security proxy can then extract the cookie and perform the appropriate account unlocking operations before sending the request to the application server. Upon receiving a positive authentication response from the application server, the security proxy can generate a cookie value and insert it as a TCP option. The user's router will then extract the cookie, store it locally, and then forward the response to the user.

5.3 Supporting Private Usernames

A token can also be user-supplied. The secret code can be shared via email, on an employee's badge, or in new employee orientation materials. When a user supplies the username for authentication, they can insert a delimiter followed by a non-public value that the proxy device can detect (e.g., `username+code`). The non-public value can be arbitrary set by the proxy administrator and changed if it was ever learned by an adversary and used in an account lockout attack. Each account should have unique secret token used for login. Since the value is only used to circumvent a lockout attack, the value could be one the user could readily access or remember, such as the user's associated employee ID or badge number. When processing authentication attempts, the proxy can search for the delimiter in a username, extract the non-public value, and verify it. The proxy can then forward the authentication request with the delimiter and code stripped from the username field for authentication by the AD server.

The user-supplied token approach has the value of being easily implemented and supported in legacy systems with the help of a proxy. When a user's account is not being attacked, they need not provide the token since the default authentication pool will be unlocked. After their account is locked due to an attack, the user only needs to type a short addition to their username to gain access. While this approach does require user training, during support calls to IT staff, the helpdesk staff can quickly remind users of the override. Finally, when setting up automated clients, such as email programs or smartphone applications, the user can choose to enter the token to ensure continued access during attacks without incurring any inconvenience. The user-supplied token avoids the complications of legacy usernames and access control that are associated with the private username approach while attaining similar benefits.

6 Evaluation of the Authentication Pools System

For our evaluation, we consider both the security effectiveness and the performance of using authentication pools. We focus on the two authentication mechanisms proposed in the preceding section: 1) authenticating requests from residential networks and 2) providing support for private usernames with tokens.

6.1 Implementation and Experimental Setup

For our baseline experiments, we configured a Windows Server 2016 Standard server to run the Active Directory service on a virtual machine with two cores and 8 GB of RAM. We configured an Exchange 2010 server on another Windows Server 2016 Standard VM with two cores and 8 GB of RAM. The Exchange server used the Active Directory server VM for authentication and the POP3 service for checking emails. To test the proposed countermeasures, we implemented two different middleboxes: the authentication pool proxy and the residential router. Our client is another Ubuntu 16.04 server VM that runs a POP3 Python script client that attempts to use our Exchange email server. While the deployed enterprise configurations will differ from our experimental setup, we believe this setup is sufficient for evaluating the security and performance characteristics.

The pool proxy implementation supports both the TLS “peek and splice” and the private username authentication mechanisms described in the preceding section. We use an Ubuntu virtual machine with 2 cores and 4096 MB of RAM and the `tcpproxy` library [13], which allows the interception and modification of packets. The `tcpproxy` library provides the functionality to wrap normal socket communication into TLS protected communication when a private key is imported. The pool proxy uses a copy of the Exchange server’s private key. The decrypted payload contains the account information. However, when `tcpproxy` uses SSL-wrapped sockets, it only provides the decrypted packet payload without network or transport layer headers. Using the `libnetfilter_queue` library and the `iptables` tool, the pool proxy intercepts the packets and extracts any TCP options from the packet headers before forwarding, to `tcpproxy` to see any token TCP options. If they are present, as shown by the green line in Figure 3, the proxy switches the request to a separate authentication pool.

To support private usernames, the pool proxy also checks each username for a + character and extracts the subsequent code. If the code is valid, the system recognizes the connection as associated with the blue line in Figure 3. When the verification of a security token succeeds, the security proxy issues commands to the AD domain controller to unlock the account if additional attempts are permitted for that pool group. After the credential verification completes at the domain controller, the security proxy sends commands to restore the account lock status. When the TCP option is used, the security proxy generates a new token and appends it to the reply packet as a TCP option. The router can thus extract the new token and store it for future use.

For the residential router, we use an Ubuntu 16.04 server VM configured with single core and 2048 MB of RAM. We then created a C program that uses the Linux `libnetfilter_queue` library and `iptables` to intercept traffic to and from the residential network. The program is designed in a fashion to allow it to be ported to commodity router hardware. The program uses the packet's destination address to determine if it is a known organizational application server. If so, supplies any associated token as a TCP option. The program also looks at the packet's source address to determine if it is an application server, and if so, the program looks for TCP options containing a token. If one is found, the router stores it for future out-bound packets and removes the option before sending the packet towards its destination. In Figure 4, we provide a diagram of this process.

6.2 Security Effectiveness

Using a methodology similar to our measurements in Section 4.2, we create a tool that emulates an attacker trying to trigger an account lockout. We use a Python script with the `poplib` library [1] to create a POP3 client. That script initiates 100 authentication attempts in rapid succession. Even with the relatively permissive NIST guidance, that volume triggered a lockout.

We unlocked the account and reset the failed attempt counter to zero. We then replicated this process using web requests to the Outlook Web Application (OWA) interface on the Exchange server manually. The outcome was the same: the legitimate user was unable to authenticate to either OWA or POP3 because the account was locked in AD.

We note that the account lockout through the OWA portal may be affected by credential caching in Microsoft's Internet Information Services (IIS). A parameter, `UserTokenTTL`, defines how long the IIS server should cache authentication tokens. The default cache flush delay is 15 minutes [20]. With that default, an attacker has 15 minutes to make unlimited password guess attempts. During that attack, the failed authentication attempts counter increases. After it hits the threshold, access via services like POP3 is denied because the account is locked, but the cached credential still allows a user to authenticate via the OWA portal. In effect, this delays the account lockout attack from affecting the OWA portal, but still allows an account lock to propagate throughout the rest of the organization. After the cache period ends, the account will also be locked on the

Table 2. Our security evaluation determined the effectiveness of the TCP option and embedded code countermeasures. Across 20 trials, both approaches correctly allow legitimate requests and deny malicious attempts.

	Router TCP Option		Username + secret	
	Token Valid	Token Invalid	Token Valid	Token Invalid
Allows access	20	0	20	0
Denies access	0	20	0	20

OWA portal. Accordingly, this caching does not ultimately affect the attack's success.

For the residential router mechanism, we primed the router by performing a legitimate authentication attempt. We then cleared the account lock status and authentication attempt counts. We then ran the attacker script without presenting a token value. However, when the legitimate user attempted to login, the router supplied the previously obtained token and the security proxy correctly unlocked the account temporarily to process the request before re-locking it. We found that the legitimate user was able to authenticate without impediment despite the ongoing attack.

We repeated the TCP option process with an adversary that tried to forge TCP tokens, by supplying random values. As expected with the low likelihood of guessing a 10-byte token value, we found that adversary never generated a correct token. The security proxy accordingly ignored these tokens and the attacker remained locked out. However, when the legitimate user attempted to authenticate, the proxy recognized the token and properly allowed the attempt. Finally, we repeated these experiments using the user-supplied tokens. The legitimate user provided a username of the format `username+token` when attempting to authenticate. In our first experiment, the attacker did not provide tokens and in our second, the attacker attempted to guess tokens randomly. As with the TCP-option experiments, we found the attacker was unable to unlock the account while the legitimate user had unimpeded access to authenticate. In Table 2, we show the numerical results of our experiments. In each experiment, we conducted 20 separate trials and the results were consistent.

6.3 Performance Evaluation

To determine the performance impact of the approach, we use end-to-end timings. We measured the amount of time required to complete an authentication request using the `time.time()` function, provided by Python, in the legitimate client's script. We measured the timing without our countermeasures implemented and with them in place. This allows us to determine the sum of all overheads present in the system. We show the results in Figure 5.

When comparing our the two authentication mechanisms, we found that the TCP option countermeasure added around 180 milliseconds of latency while the user-supplied token added 80 milliseconds of latency, compared to the same system without a countermeasure present. These overheads are so small they are

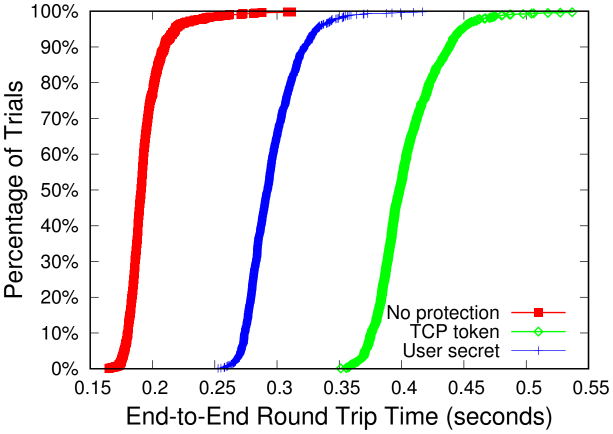

Fig. 5. We evaluate and compare end to end delay among three cases: red line shows without any middlebox or proxy, green line shows inserting tokens via TCP option and blue line shows username + secret token

unlikely to be perceived by an end-user. We believe the user secret scenario is faster in our implementation because it does not need to examine TCP options, extract tokens or append TCP tokens.

Based on these results, we find our countermeasures provide effective security benefits without introducing noticeable latency.

7 Conclusion

In this work, we explored the extent to which organizations are vulnerable to account lockouts and the impact that the lockouts could have. Looking only at the deployments of Microsoft Active Directory, we found that the majority of top companies and universities had an exposed authentication portal that would enable an attacker to launch an account lockout. Through our experiments with a partnering organization, we demonstrated the feasibility of such an attack in a production environment. We then introduced a suite of countermeasures and compared the benefits. We found that both user-supplied tokens and middlebox-added tokens would be effective and would add no perceptible delays or performance overheads.

Acknowledgements

The authors would like to thank the anonymous organization for allowing us to test our account lockout approach on their infrastructure and for providing feedback on the effectiveness of the account lockout approach when targeting different authentication portals.

This material is based upon work supported by the National Science Foundation under Grant No. 1651540.

References

1. POP3 protocol client. https://docs.python.org/3/library/poplib.html (2018)
2. 800-63B, N.S.P.: Digital identity guidelines, authentication and lifecycle management. https://pages.nist.gov/800-63-3/sp800-63b.html#throttle (2018)
3. Alsaleh, M., Mannan, M., van Oorschot, P.C.: Revisiting defenses against large-scale online password guessing attacks. IEEE Transactions on dependable and secure computing **9**(1), 128–141 (2012)
4. Aura, T., Nikander, P., Leiwo, J.: DOS-resistant authentication with client puzzles. In: International Workshop on Security Protocols. pp. 170–177. Springer (2000)
5. Cheng, Y., Chu, J., Radhakrishnan, S., Jain, A.: TCP Fast Open. https://tools.ietf.org/html/rfc7413 (2014)
6. Dean, D., Stubblefield, A.: Using client puzzles to protect TLS. In: USENIX Security Symposium. vol. 42 (2001)
7. Dufresne, J.: Python-ldap on github. https://github.com/python-ldap/python-ldap/blob/python-ldap-3.2.0/Doc/index.rst (2017)
8. Durinovic-Johri, S., Wirth, P.E.: Access control system with lockout (1997), US Patent 5,699,514
9. Eriksson, B., Barford, P., Sommers, J., Nowak, R.: A learning-based approach for IP geolocation. In: International Conference on Passive and Active Network Measurement. pp. 171–180. Springer (2010)
10. Ghasemisharif, M., Ramesh, A., Checkoway, S., Kanich, C., Polakis, J.: O single sign-off, where art thou? an empirical analysis of single sign-on account hijacking and session management on the web. In: USENIX Security Symposium. pp. 1475–1492 (2018)
11. Harvard University: Registrars of fortune 1000 companies - raw data. https://cyber.harvard.edu/archived_content/people/edelman/fortune-registrars/fortune-list.html
12. Herley, C., Florêncio, D.: Protecting financial institutions from brute-force attacks. In: IFIP International Information Security Conference. pp. 681–685. Springer (2008)
13. ickerwx: tcpproxy on github. https://github.com/ickerwx/tcpproxy (2018)
14. Ives, B., Walsh, K.R., Schneider, H.: The domino effect of password reuse. Communications of the ACM **47**(4), 75–78 (2004)
15. Koh, J.Y., Ming, J.T.C., Niyato, D.: Rate limiting client puzzle schemes for denial-of-service mitigation. In: 2013 IEEE Wireless Communications and Networking Conference (WCNC). pp. 1848–1853. IEEE (2013)
16. Liu, Y., Taylor, C.R., Shue, C.A.: Authenticating endpoints and vetting connections in residential networks. In: International Conference on Computing, Networking and Communications (ICNC) (2019)
17. Margosis, A.: Security baselines for Windows 8.1, Windows server 2012 R2 and Internet Explorer 11. https://blogs.technet.microsoft.com/secguide/2014/08/13/security-baselines-for-windows-8-1-windows-server-2012-r2-and-internet-explorer-11-final/ (2014)
18. Margosis, A.: Security baseline for Windows 10. https://blogs.technet.microsoft.com/secguide/2016/10/17/security-baseline-for-windows-10-v1607-anniversary-edition-and-windows-server-2016/ (2018)
19. Microsoft: Autodiscover for exchange. https://docs.microsoft.com/en-us/exchange/client-developer/exchange-web-services/autodiscover-for-exchange (2015)

20. Microsoft support: Changing the default interval for user tokens in IIS. https://support.microsoft.com/en-us/help/152526/changing-the-default-interval-for-user-tokens-in-iis (2018)
21. Microsoft support: Office365 login page. login.microsoftonline.com (2019)
22. MITRE Corporation: CWE-645: Overly restrictive account lockout mechanism. https://cwe.mitre.org/data/definitions/645.html (2019)
23. Monica, A.D., Baldwin, M., Cai, S., Casey, C.: Thousands of apps, one identity. https://docs.microsoft.com/en-us/enterprise-mobility-security/solutions/thousands-apps-one-identity (2016)
24. Moore, D., Shannon, C., Brown, D.J., Voelker, G.M., Savage, S.: Inferring internet denial-of-service activity. ACM Trans. on Computer Systems **24**(2), 115–139 (2006)
25. nyxgeek: Lyncsmash. https://github.com/nyxgeek/lyncsmash
26. PCIPolicyPortal: PCI compliance password requirements: Best practices to know. http://pcipolicyportal.com/blog/pci-compliance-password-requirements-best-practices-know/ (2015)
27. Pope, C., Kaur, K.: Is it human or computer? defending e-commerce with captchas. IT professional **7**(2), 43–49 (2005)
28. Pylon Technology News: Active directory in todays regulatory environment. https://pylontechnology.com/active-directory-todays-regulatory-environment/ (2014)
29. Rousskov, A.: Feature: Sslbump peek and splice. https://wiki.squid-cache.org/Features/SslPeekAndSplice (2019)
30. SANS Institute: Top 10 mistakes on windows internal networks. https://www.sans.org/reading-room/whitepapers/windows/top-10-mistakes-windows-internal-networks-1016 (2003)
31. Sherry, J., Hasan, S., Scott, C., Krishnamurthy, A., Ratnasamy, S., Sekar, V.: Making middleboxes someone else's problem: network processing as a cloud service. ACM SIGCOMM Computer Communication Review **42**(4), 13–24 (2012)
32. Standford University: Alphabetic list of us universities and domains. http://doors.stanford.edu/~sr/universities.html (1996)
33. Taylor, C.R., Guo, T., Shue, C.A., Najd, M.E.: On the feasibility of cloud-based sdn controllers for residential networks. In: IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN) (2017)
34. Taylor, C.R., Shue, C.A.: Validating security protocols with cloud-based middleboxes. In: IEEE Conference on Communications and Network Security (2016)
35. Taylor, C.R., Shue, C.A., Najd, M.E.: Whole home proxies: Bringing enterprise-grade security to residential networks. In: IEEE International Conference on Communications (ICC) (2016)
36. Wang, Y., Huang, Y., Zheng, W., Zhou, Z., Liu, D., Lu, M.: Combining convolutional neural network and self-adaptive algorithm to defeat synthetic multi-digit text-based CAPTCHA. In: IEEE International Conference on Industrial Technology (ICIT). pp. 980–985. IEEE (2017)
37. Weir, M., Aggarwal, S., Collins, M., Stern, H.: Testing metrics for password creation policies by attacking large sets of revealed passwords. In: ACM Conference on Computer and Communications Security (CCS). pp. 162–175. ACM (2010)
38. Witty, R.J., Allan, A.: Best practices in user ID formation. https://www.bus.umich.edu/kresgepublic/journals/gartner/research/117900/117943/117943.html (2003)
39. Yan, J., Blackwell, A., Anderson, R., Grant, A.: Password memorability and security: Empirical results. IEEE Security & privacy **2**(5), 25–31 (2004)