Thwarting Sophisticated Enterprise Attacks: Data-Driven Methods and Insights

Grant Ho

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2020-217
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2020/EECS-2020-217.html

December 18, 2020

Thwarting Sophisticated Enterprise Attacks: Data-Driven Methods and Insights

by

Grant Ho

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Vern Paxson, Co-chair
Professor David Wagner, Co-chair
Professor David Bamman

Fall 2020

Thwarting Sophisticated Enterprise Attacks: Data-Driven Methods and Insights

Abstract

Thwarting Sophisticated Enterprise Attacks: Data-Driven Methods and Insights

by

Grant Ho

Doctor of Philosophy in Computer Science

University of California, Berkeley

Professor Vern Paxson, Co-chair

Professor David Wagner, Co-chair

This dissertation builds new defenses to thwart digital attacks on enterprises. Specifically, we develop a set of data-driven insights and methods that enable organizations to uncover and stymie three prominent enterprise attacks: spearphishing, lateral phishing, and lateral movement. For each of these threats, we present new conceptual models that deconstruct each attack into a set of fundamental actions that an attacker must perform in order to succeed, enabling organizations to more precisely search for signs of such malicious activity. The successful detection systems we construct based on these models highlight the value of decomposing and pursuing attacks along two facets: preventing attackers from gaining entry into an enterprise's network and hunting for attacker activity within an organization's internal environment.

Even with a clear specification of what to look for, uncovering sophisticated attacks has long eluded enterprises because such attacks give rise to a detection problem with two challenging constraints: an extreme class imbalance and a lack of ground truth. In particular, targeted enterprise attacks occur at a low rate, reflect the work of stealthy attackers (and thus frequently remain unknown and unlabeled), and transpire amidst a sea of anomalous-but-benign activity that inherently occurs within modern enterprise networks. This setting poses fundamental challenges to traditional machine learning methods, causing them to detect an insufficient number of attacks or produce an intractable volume of false positives. To overcome these challenges, we present a new approach to anomaly detection for security settings, *specification-based anomaly detection*, which we use to construct new detection algorithms for identifying rare attacks in large, unlabeled datasets.

Combining these algorithms with the attack models we develop, we design and implement a set of detection systems that collectively form a defense-in-depth approach to unearthing and mitigating enterprise attacks. Through collaborations with three large organizations, we validate the efficacy and practicality of our approach. Given the ability of our systems to detect a wide-range of attacks, the low volume of false positives they generate, and the real-world adoption of many of our ideas, this dissertation illustrates the utility and promise of a data-empowered approach to thwarting enterprise attacks.

Contents

Acknowledgments

A wonderful group of friends, colleagues, mentors, and family members have helped me do great work over the course of my Ph.D. and have filled these past few years with happy memories.

I could not have been more blessed to have two outstanding advisors: Vern Paxson and David Wagner. Like all good advisors, both of you provided abundant support and encouragement: from venting with me about Reviewer #2, to brainstorming how we should overcome a technical hurdle, to diving into a cool new finding, I left every one of our discussions with a sense of clarity and excitement. At the same time, you never shied away from constructively saying exactly what you thought, providing me with a fountain of sage advice, rebuttals to flawed approaches, and wonderful extensions to our ideas. Even though I did not heed all of your advice and suggestions (often to my detriment), I appreciate the freedom you gave me to chart my own course and uncover new results that sometimes surprised us all. When I become a professor, I know I will draw heavily from our advising relationship; and hopefully I can guide my students through an equally inspiring and enriching experience.

In addition to my formal advisors, I had the privilege to work with two other incredible mentors: Stefan Savage and Geoff Voelker. Working with the two of you during the second half of my Ph.D. was one of the best decisions I ever made in graduate school. I have grown tremendously from the technical feedback and career advice you both have provided. The animated discussions that I had with you and my advisors remain some of my favorite memories of graduate school and helped hone my taste for good research. Additionally, I want to thank Dan Boneh for serving as my first research mentor during my undergraduate years. Your enthusiasm for security and research, and the guidance you provided throughout our work, helped launch my research career. Collectively, the five of you have pushed me to become an ambitious researcher with a high standard for clarity, impact, and rigor.

Several additional faculty also provided me with support and insightful feedback over the course of my Ph.D. Dawn Song helped kick-start and refine my early work on IoT security with a number of valuable ideas and discussions about my early papers. Raluca Ada Popa gave me insights into new areas of crypto-systems research and warmly welcomed me into her group's social and research events. Aside from his technical knowledge, Alex Snoeren is simply magical when it comes to travel logistics; thank you for answering my panicked airport phone call, after I missed an international flight by minutes, and helping me find a crazy multi-hop route that got me to my conference talk on time. I am also grateful to David Bamman and Joey Gonzalez for taking the time to serve on my dissertation committees, and for their helpful comments on my work.

All of my research benefited from the ideas and work of many colleagues and co-authors: Aashish Sharma, Asaf Cidon, Ashkan Hosseini, Chris Grier, Damon McCoy, Derek Leung, Devdatta Akhawe, Kurt Thomas, Lior Gavish, Lucas Ballard, Marco Schweighauser, Mayank Dhiman, Mobin Javed, Mohamed Ibrahim, Moheeb Rajab, Neil Shah, Niels Provos, and Pratyush Mishra. A special thank you to Neil, Ashkan, Derek, and Pratyush for giving me the opportunity to mentor and advise you during parts of your career.

Collaborations with industry have led to some of my most exciting research papers. One piece of advice I would give to other researchers is to seek out and work with industry colleagues whom

you respect and who appreciate the value of research. Thank you to Aashish Sharma, Asaf Cidon, Brad Miller, Devdatta Akhawe Lior Gavish, Lucas Ballard, Marco Schweighauser, Mayank Dhiman, Moheeb Rajab, and Niels Provos for providing me with invaluable connections to exciting work in industry. Beyond your technical contributions to the work we did, your help in advocating internally for our research created opportunities that may never have existed otherwise.

Thank you to the incredible administrators and staff at Berkeley: Angie Abbatecola, Jean Nguyen, Jon Kuroda, Lena Lau-Stewart, and Shirley Salanio. You guided me through a labyrinth of logistical hurdles and kept our infrastructure running; an extra big thank you to Angie and Lena for all of your help organizing a potpourri of seminars and social gatherings for the security group. Your generosity and time made my Ph.D. a fun and well-supported experience, and I know that is true for the broader security group at Berkeley as well.

When I was deciding where to go for graduate school, people often told me that my advisor relationship would be most important part of my Ph.D. experience. While that has largely been true, for myself and many other students I know, my fellow students and labmates played an equally important part. I am so grateful to have met a wonderful group of colleagues and friends (not mutually exclusive) over the past few years: Ariana Mirian, Austin Murdock, Brian Kilberg, Chelsea Finn, Chris Grier, Elissa Redmiles, Frank Li, Greg Kahn, Henry Cook, Henry Corrigan-Gibbs, Kurt Thomas, Marcell Vasquez-Chanlatte, Marissa Ramirez de Chanlatte, Max Curran, Michael Armburst, Nathan Malkin, Nicholas Carlini, Paul Pearce, Pratyush Mishra, Richard Shin, Rishabh Poddar, Sandy Huang, Wenting Zheng, Zakir Durumeric, and all the wonderful peers who made my Ph.D. a fantastic experience. From Barcelona to Tokyo and board games to potlucks, you have filled my Ph.D. with fun and laughter; and I have learned a lot about research and life through our time together. I can't wait to see the things we do and trips we have together in the future. And of course, I owe an extra-special thank you to my amazing partner, Eric Moorman, for your unwavering love and support.

Finally, I dedicate this dissertation to my family: my brother, Albert Ho, and my parents, Xudong He and Yingjia Ding. Because of my parents' hard work, I always had the resources and opportunities to pursue whatever goals I wanted in life. Thank you to all three of you for your many years of love, support, and belief in me.

Chapter 1

Introduction

Digital attacks on enterprises remain an unsolved and potent threat to society. Over the past decade, organizations spanning every economic sector and governments across the world have fallen victim to a deluge of attacks [10, 24, 66, 118, 121]. In many of these breaches, attackers successfully stole sensitive data about millions of people, ranging from healthcare records to government security clearance reports, for the purposes of financial gain and espionage [35, 41, 78]. Recently, such attacks have become seemingly more audacious and visible, with state-sponsored attackers openly weaponizing and leaking stolen data from political figures and organizations in attempts to influence the outcome of national elections, such as the 2016 US Presidential Election and the 2017 Presidential Election in France [121]. Equally troubling, these attacks continue to evolve and expand, as seen in the wave of ransomware-style attacks that monetize their compromise by extorting users and organizations [92]. Not only do these newer attacks exact financial harm on their victims, but they often cause physical damage from the critical infrastructure they disable over the course of an attack [16, 92, 96]; indeed, hospital patients have died as a result of the disruption caused by ransomware attacks [122]. Going beyond incidental damage, a number of nation-state operated attacks have deliberately and successfully sought to inflict tangible harm, such as the destructive 2014 attack on Sony Pictures by North Korea and Russian cyberattacks that targeted Ukraine's power grid [66, 91]. Collectively, the billions of dollars in financial loss, concrete harm to individuals and geopolitical stability, and growing litany of successful attacks illustrate that enterprises struggle to defend themselves against modern-day attacks.

At a high-level, enterprise attacks involve two key stages: the actions an attacker takes to gain entry into an organization's network, and the subsequent activity they perform within the enterprise's internal environment to achieve their ultimate goal. Traditionally, the defenses that enterprises have deployed focus on the first aspect: hardening an organization's perimeter to prevent an attacker from establishing internal access. These defenses, ranging from network firewalls, antivirus software, hardened web browsers, and automatic updates, effectively reduce an organization's accessible attack surface, and have nearly eliminated once-ubiquitous threats such as massively-propagating worms and drive-by downloads.

Unfortunately, modern-day attacks reflect the work of tenacious adversaries who have adapted to these defenses by expanding their exploit strategies. In recent years, attackers have shifted from

using technical attacks that abuse weaknesses in code and machines to deploying *social engineering attacks* that target humans with sophisticated lies and deception. For example, spearphishing, the most prominent of these social engineering attacks, involves an attacker cleverly crafting a deceptive email to one or more employees at an organization. These malicious emails ultimately trick their victims into performing a dangerous action that enables the attacker to establish a foothold behind the wall of defenses that organizations deploy at their borders. Myriad successful attacks over the past decade have involved the use of spearphishing [118]: the theft of 21.5 million sensitive background check records from the US Office of Personnel and Management (OPM) in 2015, the breach of Anthem Healthcare that resulted in 37.5 million stolen healthcare records, and the compromise and politically-calculated release of emails from the Democratic National Committee and John Podesta in 2016 all involved successful spearphishing attacks [41, 78, 121]. These types of social engineering attacks remain difficult to defend against because they rely solely on human deception and manipulation, enabling attackers to bypass the many technical defenses the community has developed.

With this new human-centered means of compromise, attackers routinely gain access to a legitimate employee's credentials and/or machine. In some cases, this initial access provides attackers with all they need to accomplish their objective. However, in many cases, the initial machine that an attacker has compromised does not enable them to fulfill their goal because they ultimately seek access to more sensitive data on internal servers or wish to spread onto as many internal machines as they can. Thus, as part of their attacks, adversaries frequently engage in a process known as *lateral movement*: using the legitimate access provided by the initial user(s) and machine(s) they have compromised to move onto other accounts and machines in an enterprise's network until they reach their ultimate target [14, 78, 115]. Unfortunately, thwarting lateral movement, this more advanced stage of modern attacks, remains an understudied problem because researchers have lacked access to sufficiently rich and realistic data, and because defenses have traditionally focused on securing an enterprise's network perimeter and endpoint machines. Coupled with the messy complexity and scale of most enterprises, this dearth of practical techniques and insights leaves many organizations unable to effectively understand the security and nuances of their internal landscape. As a result, once attackers have compromised a legitimate enterprise machine, they can often exercise the inherent access and capabilities on their initial foothold to spread unimpeded throughout an organization's internal environment in pursuit of sensitive data or destructive havoc.

This dissertation presents a promising path forward to mitigating these sophisticated enterprise attacks. We develop a practical set of data-driven methods and insights that organizations can use to thwart an attack across its entire life-cycle. First, we examine new methods that enable organizations to detect and repel attacks at their perimeters. In this first part, we specifically focus on methods to detect spearphishing attacks, the predominant method that attackers use to breach an organization's perimeter. We present a conceptual framework for characterizing spearphishing attacks and a new anomaly detection algorithm that achieves orders-of-magnitude better performance than traditional machine learning methods. Second, we explore new paradigms and techniques for surfacing attacker activity within an enterprise's internal environment. Namely, assuming an attacker can successfully breach an organization's perimeter defenses, can organizations still thwart the attack before it successfully achieves its malicious goal? We affirmatively answer this ques-

tion by presenting empirically-backed algorithms to detect and hinder an attacker from spreading beyond their initial point of compromise to other machines or accounts within an enterprise. Ultimately, the methods, models, and insights we develop can significantly improve an enterprise's security against sophisticated attacks, while incurring a low and practical burden on an organization's security team.

Primary Contributions

This dissertation presents three primary contributions that improve an organization's ability to mitigate sophisticated attacks. First, we introduce a new approach to uncovering rare, unlabeled attacks in large, complex datasets. Leveraging this approach, we construct two novel detection systems for identifying sophisticated attacks with a practical volume of false positives in real-world enterprise networks. Second, this dissertation develops new mental models that advance our understanding and ability to detect enterprise phishing attacks, marking a practical step forward in thwarting one of the most prevalent and successful attack methods used by adversaries of all skill sets. Finally, we present a new set of empirical findings and detection algorithms that can successfully mitigate attacks even after an adversary has successfully established a foothold within an organization's internal environment. Taken together, these last two contributions highlight a powerful approach to applying decomposition to security problems: by deconstructing attacks into a series of fundamental stages and actions, defenders can construct systems that more precisely target key elements of an attack and then combine them for a stronger defense-in-depth approach to security.

Specification-Based Anomaly Detection: A New Paradigm for Uncovering Sophisticated Attacks

With the advent of neural networks and modern machine learning techniques, a data-driven approach for detecting enterprise attacks might seem straightforward: extract a set of features from an enterprise's logging data and apply a state-of-the-art deep learning algorithm to find new instances of an attack. While appealing, this dissertation illustrates that such "straightforward" machine learning approaches encounter three critical challenges that make them difficult to deploy in our security settings.

Challenges in Detecting Sophisticated Attacks: First, enterprises face a perplexing data paradox when it comes to defending against sophisticated attacks: although each attack causes tremendous damage and many organizations fall victim to such attacks, on a per-organization level, most enterprises only encounter a small number of such attacks (e.g., on the order of one attack every several months or even years). This low base-rate frequency and the resulting intractable class imbalance prevent traditional supervised machine learning algorithms from learning a meaningful attack representation or classification boundary, resulting in orders-of-magnitude too many false positives or poor detection performance. Second, because these attacks are sophisticated and naturally stealthy, most enterprises lack any ground truth knowledge about attacks in their data, even

if they have been compromised. Although many applications of machine learning encounter unlabeled datasets, the typical solutions used in these situations, such as augmenting one's training data with labeled examples from external sources, face significant obstacles in the settings we study. Because of the targeted (tailored) nature of many enterprise attacks and the sensitive details that sharing concrete data about such incidents might reveal, practitioners often lack effective methods to obtain externally labeled attack data that generalizes across multiple organizations. Third, the complexity and scale of modern enterprises inherently produces a deluge of diverse behavior and events, resulting in an abundance of anomalous-but-benign activity. This means that a traditional detection approach, such as labeling anomalous events as potential attacks, will ultimately produce an untenable number of false alerts.

Our Approach: To address these challenges, we develop a new approach for detecting attacks: specification-based anomaly detection, which we subsequently use to develop practical detection systems under these extreme constraints. Given the lack of labeled data and the difficulty in obtaining a sufficiently large number of labeled attack events, unsupervised learning and anomaly detection methods reflect the most promising approaches in the suite of traditional machine learning methods. Unfortunately, as discussed above, the low base-rate for attacks and plethora of anomalous-but-benign events inherent within large enterprises cause traditional anomaly detection methods to generate too many false positives.

Our approach, specification-based anomaly detection, reduces this volume of false positives by redesigning anomaly detection algorithms to look for the most "suspicious" events, rather than the most statistically anomalous events. First, specification-based anomaly detection leverages a user-provided "attack specification" that defines the fundamental stages or characteristics of an attack. A core tenet of security is the notion of a *threat-model*: a well-defined specification for what constitutes an attack and the constraints under which attackers and defenders operate. As we show through two different detection systems in this dissertation, well-defined threat models for an attack often lead to a natural and apt specification for our detection algorithms to use.

Intuitively, an attack specification provides a multidimensional characterization of how "suspicious" an event is: events that strongly exhibit many of an attack's fundamental characteristics are more likely to reflect an attack. Leveraging this specification and intuition, anomaly detection algorithms within our framework identify attacks by surfacing events that exhibit many of the fundamental characteristics of an attack. In practice, this often corresponds to identifying events that both contain many specified attack properties and deviate significantly from benign data. By incorporating domain knowledge to look for events that have the most suspicious features, rather than those that just have the most statistically-anomalous features, our paradigm leads to detection algorithms that achieve orders-of-magnitude fewer false positives than traditional machine learning methods, which we illustrate in Chapters 3 and 6.

New Insights into Enterprise Phishing Attacks

As attackers have shifted away from exploiting technical vulnerabilities to leveraging deception and social-engineering attacks to fool unsuspecting victims, existing defenses have struggled to adapt and mitigate these new types of attacks. To remedy this gap, this dissertation develops a new

conceptual framework that deconstructs phishing attacks, the predominant method of compromise used by modern-day adversaries [24, 105], into a set of fundamental stages that any successful phishing attack must execute. Leveraging this model of phishing attacks, in conjunction with a new specification-based anomaly detection algorithm, we present practical detection systems for uncovering this sophisticated form of social-engineering. Extending this work, we also conduct a large-scale empirical study spanning over 90 real-world organizations to understand the nature of newer forms of phishing attacks that adversaries might use to spread within an enterprise's internal environment. Our study suggests that many organizations already suffer from this newer form of phishing, but that these attacks do not yet exhibit the stealthiness and sophistication associated with more traditional spearphishing attacks. This suggests that while adversaries have significant room to improve their attacks' efficacy in the future, organizations can reap immediate defensive benefits from the practical detection methods that we propose.

New Methods for Defending-in-Depth against Sophisticated Attacks

Traditionally, security defenses have focused on preventing attacks at organization's perimeter and endpoints. We present some of the first large-scale results that illustrate the feasibility and utility of mitigating an attack by hunting for the *internal movement* activity that adversaries frequently make *within* an enterprise. Adopting this paradigm enables organizations to thwart an attack even after an adversary has managed to bypass an enterprise's perimeter and endpoint defenses. Specifically, we leverage two real-world datasets to examine two ways in which an attacker might use their initial access to spread to other internal accounts and machines within an organization. Through these studies, we develop new empirical insights and detection algorithms that can identify when an attacker attempts to expand beyond their initial foothold, enabling organizations to restrict an attacker's capabilities and prevent adversaries from successfully executing their attacks.

Outline

This dissertation consists of two parts, each corresponding to a set of insights and methods that organizations can use to mitigate different stages of modern attacks. In Part I, we develop methods to prevent attackers from gaining access to an organization's internal environment. Subsequently, in Part II we explore the feasibility of thwarting an attack, even after an adversary has managed to compromise an employee or machine within an organization.

We begin Part I with an overview of phishing attacks and defenses. To address this unsolved threat, we present a system for detecting credential spearphishing in Chapter 3 that combines a new conceptual framework for modeling phishing attacks along with our novel specification-based anomaly detection framework.

Next, Chapter 4 in Part II provides an overview of prior work that aims to hinder an attacker who has managed to breach an organization's perimeter defenses. In Chapter 5, we explore the evolving nature of enterprise phishing attacks to better understand and mitigate a new form of phishing that aims to spread from an already compromised enterprise account to additional accounts within an organization. In Chapter 6 we then present a practical detection system for lateral

movement attacks, which draws upon our framework for specification-based anomaly detection to achieve a low, practical volume of false positives.

We conclude in Chapter 7 with a reflection of this dissertation's contributions and a discussion of areas for future work.

Practical Impact: Taken together, the insights, algorithms, and systems developed in this dissertation help pave an exciting path forward for defending-in-depth against the many successful and sophisticated attacks that enterprises face. The conceptual and algorithmic frameworks developed in Part I provide a practical approach that organizations can use to mitigate attacks at their perimeter; government institutions, such as the Lawrence Berkeley National Laboratory, and large enterprises, such as Facebook, have implemented detection systems based on our work and have used them to identify real-world spearphishing attacks. The novel insights we developed about lateral phishing attacks in Chapter 5 has led to new detection initiatives and improvements in Barracuda Network's commercial anti-phishing services, which protect millions of enterprise users. Finally, the system we developed in Chapter 6 for detecting lateral movement attacks has helped Dropbox improve the security posture of their environment, and a version of our system is currently being implemented for production use at Dropbox to help secure their networks against real-world attackers.

Part I

Thwarting Attacks at an Enterprise's Perimeter

Chapter 2

Perimeter Defenses: Introduction and Related Work

2.1 Introduction

In the early years of computer security, the poor software security and the lack of basic network hygiene enabled attackers to compromise organizations via simple vulnerabilities. Today, a number of practical defenses, such as network firewalls and automatic updates, make such technical exploits increasingly difficult. In response, attackers have evolved and now draw upon a new set of compromise methods known as *social engineering*. Rather than targeting technical vulnerabilities, social engineering attacks aim to trick their victims into performing dangerous actions on behalf of the attacker through cleverly-crafted and deceptive content.

Part I of this dissertation explores a new approach to mitigating spearphishing, the most prevalent form of social engineering attacks on enterprises [24, 105]. In a spearphishing attack, the adversary sends a tailored and deceptive email that convinces its recipient to perform a dangerous action, such as entering their password on a malicious website or downloading a malware-infected document. This malicious action, performed by a legitimate but unsuspecting user, enables the attacker to gain access to the internal environment and assets of an organization. Because phishing attacks exploit human confusion and mistakes, they allow attackers to bypass the need for technical vulnerabilities, and thus prove difficult to mitigate via traditional software and systems security approaches. As a result, modern-day attackers routinely turn to phishing attacks as their method of choice for compromising high-value targets: the US State Department, the Pentagon, Google, RSA, Anthem Healthcare, the Democratic National Committee, and numerous other organizations have all fall victim to successful phishing attacks [11, 24, 121].

Although the community has proposed a range of defenses against phishing attacks, these defenses face a number of practical limitations, ranging from usability and deployment challenges to detection techniques that generate intractable volumes of false positives and thus provide insufficient protection against more targeted forms of phishing. To address this defensive gap, we propose a new approach to detecting spearphishing attacks that leverages two novel contributions.

First, we develop a new conceptual framework that deconstructs phishing attacks into two key stages, which all successful attacks must perform. This framework allows us to derive a set of salient features that aptly characterize the suspiciousness of an email. Next, we present a new anomaly detection algorithm, direct anomaly scoring (DAS), which we combine with our feature set. Ultimately, these two contributions lead to a detection system that can identify a variety of real-world spearphishing attacks with a low, practical volume of false positives.

In the remainder of this chapter, we provide an overview of prior work on defending against phishing attacks. Subsequently, in Chapter 3, we describe and evaluate our new approach to detecting spearphishing attacks. Chapter 3 adapts work that was published at the 2017 Usenix Security Symposium [49], where it won a Distinguished Paper Award and the 2017 Internet Defense Prize.

2.2 Related Work and Background

Prior work on phishing attack falls into three broad defensive approaches: stronger authentication and verification mechanisms, security training that improves a user's ability to independently spot and avoid phishing attacks, and techniques to detect and block phishing attacks.

Stronger Identity Verification Frameworks

Phishing attacks exploit the weak notions of authenticity that exist in computer systems. Attackers attempt to fool the receipt of a phishing email by masquerading as a sender that the receipt trusts. Prior work has pursued two ways to alleviate this problem: better mechanisms to verify the sender of an email, and enhanced authentication protocols that verify users in a phishing-resistant manner.

Email Sender Verification

To provide better assurances about the authenticity of an email's sender, the community has proposed a set of three email authentication mechanisms over the past several decades. The Sender Policy Framework (SPF) enables organizations to list a set of IP addresses and/or hostnames in their organization's DNS metadata that specify which servers can legitimately send email on behalf of the organization and its users [128]. A recipient's mail server can then use this verification metadata to check whether an email purportedly from an organization came from an authorized outbound mail server. DKIM—Domain-Key Identified Mail [124]—expands upon the idea of SPF by providing a cryptographic signature in an email's header that a recipient can use to ensure that the email truly came from its purported sender's domain. To round out and improve the deployment of these protocols, a consortium of industry partners developed the DMARC framework (Domain-based Message Authentication, Reporting & Conformance) [88], which details a policy for effectively using an email's SPF and DKIM information, handling a variety of edge cases, and providing error and diagnostics information to an email's sending domain.

Although these frameworks help mitigate phishing attacks, they suffer from a number of fundamental and real-world limitations. First, all of these frameworks depend on the legitimate

email sender's domain to configure and properly maintain SPF or DKIM records; without this active cooperation, email recipients cannot use these frameworks to validate whether an email truly came from the sender's organization. Second, many phishing attacks exploit a gap in an email's perceived sender and its (potentially) verified sender, which these frameworks cannot defend against [81, 130]. For example, an attacker trying to impersonate an employ from "example.com", might instead create a domain "exampIe.com" (replacing a lowercase "L" with an uppercase "I"). The attacker could then configure their look-alike domain with the appropriate SPF and DKIM information, and use this deceptive domain to execute their phishing attacks. Despite their visual similarity, existing frameworks like DMARC, SPF, and DKIM will mark the phishing emails from "exampIe.com" as legitimate and authorized emails. However, an unsuspecting user might not realize that the malicious emails sent from an email address at "exampIe.com" come from an inauthentic sender, leaving the user vulnerable to the attack, despite all of these verification protocols working correctly. Finally, recent work has illustrated that the practical complexity of email delivery and verification introduces a number of real-world vulnerabilities that attackers can use to fully bypass these verification protocols across a range of popular email service providers [19, 52].

Multi-Factor Authentication

To help mitigate the damage caused by a successful phishing attack, the security community has developed "multi-factor authentication" (MFA) protocols. In MFA-based authentication, not only do users need to provide a traditional form of verification, such as a password, but they also need to verify their identity via additional method(s), such as a one-time generated pass-code sent to their mobile device. Intuitively, this approach aims to mitigate phishing attacks by requiring an attacker to compromise two or more sources of authentication for their attack to succeed.

Although this approach provides significant defensive value against phishing attacks, it has long suffered from real-world usability and deployment issues that remain today [31, 103]. For example, if a user loses their second-factor authentication device (e.g., a broken authentication token or a stolen phone), then they risk losing access to any resources and data shielded by MFA-based authentication. Additionally, modern attacks have adapted to circumvent many of the most popular and widely-accessible forms of multi-factor authentication [26, 99, 113], by explicitly stealing a user's second-factor information (e.g., one-time password) in addition to their password as part of the phishing attack.

User Security Training

Because phishing attacks rely on deceiving their victims, prior work has studied how we can educate users to better recognize these attacks, as well as the underlying human factors that lead users to fall victim to online deception. Collectively, these prior studies suggest that some demographic factors, such as age and gender, might contribute to a user's susceptibility to phishing attacks [86]. Based on these results, prior work suggests that tailoring interventions and anti-phishing education towards different demographic groups might improve their overall efficacy. Other studies indicate

that the loose nature of trust on the web inherently leads to successful phishing attacks because users do not have simple and effective means to validate a website or user's true identity [55, 108]. Amidst these different findings, prior work has shown that some user education efforts decrease the efficacy of phishing attacks; however, these studies consistently report that a substantial fraction of users nonetheless fall victim to these deceptive attacks, even after users undergo anti-phishing training [61, 106, 109].

Detecting Phishing Attacks

Given the limitations of verification mechanisms and users' abilities to independently identify phishing attacks, a long line of prior work has proposed methods to systematically detect phishing attacks. This work pursues two primary directions: identifying malicious websites that host phishing content, and identifying phishing emails that users receive.

Detecting Phishing Webpages and URLs

Several prior systems aim to identify phishing websites and URLs, with the goal of adding these sites to widely distributed blocklists that prevent a user from loading malicious content. Garrera et al. [37] develop a logistic regression classifier that extracts a set of features from a URL, such as its domain's registration information, the presence of certain keywords, and signs of domain obfuscation, to determine whether the URL points to a phishing website. Similarly, PhishNet [95] attempts to identify new phishing URLs from a corpus of malicious URLs by using natural language techniques to compute whether a new URL has a close lexicographical similarity to any known phishing URL. While these two approaches rely solely on properties of a phishing website's URL, other work has also explored incorporating content from a URL's webpage into their classification decision [110, 123, 134]. These systems leverage a webpage's content, such as text from the rendered HTML DOM and/or images displayed on the website, to extract a set of features that their classifiers use in conjunction with URL-based features to label a website as phishing or not. Newer work has also proposed methods that allow organizations to detect when phishing websites impersonate them, by analyzing which external domains fetch resources or redirect users to their websites [84].

Although these prior works provide some protection against phishing attacks, and have seen practical deployment in large-scale systems such as Google SafeBrowsing, they defend primarily against mass phishing attacks that reuse attack infrastructure across many different victims and/or impersonate a handful of popular brands. Against more sophisticated attacks, such as targeted spearphishing emails sent to a particular organization, these defenses are unlikely to succeed. Additionally, many of these approaches rely on the use of globally deployed blocklists of malicious URLs, which face a number of challenges with respect to adversarial evasion, such as malicious website cloaking and the rapid churn in phishing infrastructure and content [83, 110, 133].

Detecting Phishing Emails

Because many enterprise users encounter phishing attacks via a malicious email, the work in this dissertation, as well as some prior work, has explored methods for detecting phishing emails [5,29, 34,38,60,114,135]. At a high-level, prior approaches build behavioral models for a sender's typical email by constructing a set of features based on an email's metadata, stylometry, and timing. These prior methods then classify an email as phishing or not by using these behavioral models to check whether a new email's features differ from the sender's historical profile. Unfortunately, these prior approaches cannot detect a variety of phishing attacks, such as those sent by an externally-spoofed email address with no prior history (and thus no behavioral model to compare the attack email against).

Equally important, the detection algorithms proposed by prior work generate too many false positives for practical use, suffering from a recurring challenge encountered throughout this dissertation: the base rate fallacy [4]. Because sophisticated attacks, such as spearphishing, occur relatively infrequently and because the volume of benign events (emails) at enterprises is incredibly large, even a low false positive or error rate will generate too many alarms for practical use. For example, IdentityMailer [114] achieves false positive rates in the range of 1–10%. Although quite low, on the real-world dataset we drawn upon in Chapter 3, which contains over 250,000 emails per day, a false positive rate of 1% would lead to a daily volume of over 2,500 false alarms.

Recently, new forms of phishing have emerged that do not aim to compromise any user's credentials or systems within an enterprise. Rather, these attacks aim to trick a user into performing a fraudulent financial transaction for the attacker. In these types of phishing attacks, known as business email compromise (BEC), an adversary masquerades as a legitimate business partner and convinces the phishing email's recipient to transfer a substantial sum of money to the attacker. To mitigate this threat, recent work has explored developing classifiers based on natural language processing techniques to detect these types of phishing emails [21]. In this dissertation, we do not study this type of transaction-based phishing. Instead, we focus on attacks that seek to gain access to an enterprise's internal environment and data, which remain the goal of many sophisticated attacks [11, 24, 41, 72, 78, 101, 107, 118, 121].

Chapter 3

Detecting Credential Spearphishing Attacks

3.1 Introduction

Over the past several years, numerous high-profile breaches have highlighted the growing prevalence and potency of spearphishing attacks. Leveraging these attacks, adversaries have successfully compromised a wide range of government systems (e.g., the US State Department and the White House [24]), prominent companies (e.g., Google and RSA [11]), and recently, political figures and organizations (e.g., John Podesta and the DNC [121]).

Spearphishing attacks take several forms. Historically, some of the most well-known attacks involve an email that tries to fool the recipient into downloading a malicious attachment. However, in our work, which draws upon several years' worth of data from the Lawrence Berkeley National Lab (LBNL), a large national lab supported by the US Department of Energy, none of the successful spearphishing attacks involved a malicious attachment. Instead, the predominant form of spearphishing that LBNL encounters is *credential spearphishing*, where a malicious email convinces the recipient to click on a link and then enter their credentials on the resulting webpage. For an attachment-driven spearphishing attack to succeed against a site like LBNL, which aggressively scans emails for malware, maintains frequently updated machines, and has a team of several full-time security staff members, an attacker will often need to resort to an expensive zero-day exploit. In contrast, credential spearphishing has an incredibly low barrier to entry: an attacker only needs to host a website and craft a deceptive email for the attack to succeed. Moreover, with widespread usage of remote desktops, VPN applications, and cloud-based email providers, stolen credentials often provide attackers with rich information and capabilities. Thus, although other forms of spearphishing constitute an important threat, credential spearphishing poses a major and unsolved threat in-and-of itself.

This chapter presents a new approach for detecting credential spearphishing attacks in enterprise settings. Uncovering these attacks proves highly challenging due to base-rate issues. For example, our enterprise dataset contains 370 million emails, but fewer than 10 known instances of spearphishing. Consequently, many natural methods fail because their false positive rates are too high: even a false positive rate as low as 0.1% would lead to 370,000 false alarms. Addi-

tionally, with such a small number of known spearphishing instances, standard machine learning approaches seem unlikely to succeed: the training set is too small and the class imbalance too extreme.

To overcome these challenges, we introduce two key contributions. First, we present an analysis of characteristics that we argue are fundamental to spearphishing attacks; from this analysis, we formulate a new attack specification and corresponding set of features that target the different stages of a successful spearphishing attack. Second, we introduce *DAS*, a simple, new anomaly detection technique that requires no labeled training data and operates in a non-parametric fashion. Our technique allows its user to easily incorporate domain knowledge about their problem space into the anomaly scores DAS assigns to events. As such, in our setting, DAS can achieve an order-of-magnitude better performance than standard anomaly detection techniques that use the same features. Combining these two ideas together, we present the design of a real-time detector for credential spearphishing attacks.

Working with the security team at LBNL, we evaluated our detector on nearly 4 years of email data (about 370 million emails), as well as associated HTTP logs. On this large-scale dataset, our detector generated an average of under 10 alerts per day; and on average, an analyst can process an entire month's worth of these alerts in 15 minutes. Assessing our detector's true positive accuracy, we found that it not only detected all-but-one spearphishing attack known to LBNL, but also uncovered 2 *previously undiscovered* spearphishing attacks. Ultimately, our detector's ability to identify both known and novel attacks, and the low volume and burden of alerts it imposes, suggests that our approach provides a practical path towards detecting credential spearphishing attacks.

3.2 Attack Taxonomy and Security Model

In a spearphishing attack, the adversary sends a targeted email designed to trick the recipient into performing a dangerous action. Whereas regular phishing emails primarily aim to make money by deceiving any arbitrary user [108, 123], spearphishing attacks are *specifically targeted* at users who possess some kind of *privileged access or capability* that the adversary seeks. This selective targeting and motivation delineates spearphishing (our work's focus) from regular phishing attacks.

Taxonomy for Spearphishing Attacks

Spearphishing attacks leverage a wide range of deceptive strategies and content. To better understand this complex problem space, we present a taxonomy that characterizes spearphishing attacks across two dimensions: the *lure* and the *exploit* used by an attack. These dimensions correspond to the two key stages of a successful attack. Throughout this chapter, we refer to the attacker as Mallory and the victim as Alice.

Figure 3.1: Examples of four different impersonation models for a real user “Alice Good”. Our work focuses on detecting the latter three threat models, as discussed in Section 3.2: *name spoofer*, *previously unseen attacker*, and *lateral attacker*.

Lure

Spearphishing attacks require Mallory to convince Alice to perform some action described in the email. To accomplish this, Mallory needs to imbue her email with a sense of trust or authority that convinces Alice to execute the action. Attackers typically achieve this by sending the email under the identity of a trusted or authoritative entity and then including some compelling content in the email.

Impersonation Model: Spearphishing involves impersonating the identity of someone else, both to create trust in the recipient and also to to minimize the risk of attribution and punishment. There are several types of impersonation (illustrated in Figure 3.1):

1. An *address spoofer* crafts their attack email such that it displays the email address of a trusted individual in its `From` header. The attacker may spoof the name in the `From` header as well, so that the attacker's `From` header exactly matches the true user's typical `From` header. DKIM and DMARC [88] block this impersonation model by allowing domains to sign their sent emails' headers with a cryptographic signature, which receiving servers can verify with a DNS-based verification key. In recent years, these protocols have seen increasingly widespread adoption, with many large email providers, such as Gmail, deploying them in response to the rise of phishing attacks [13].
2. A *name spoofer* forges the name in their email's `From` header to exactly match the name of an existing, trusted individual (e.g., `Alice Good` in “Alice Good <alice@evil.com>”). However, in this impersonation model, the attacker does not forge the email address of their

`From` header, relying instead on the recipient to only view the name of the sender, or on the recipient's mail client to show only the name of the sender. By not spoofing the `From` email address, this impersonation model circumvents protocols like DKIM and DMARC.

3. A *previously unseen attacker* selects a name and email address to put in the `From` field of the spearphishing email that the recipient might perceive as trustworthy or similar to a real user's identity; however, neither the name nor the email address actually match a true user's name or email address. For instance, Mallory might choose to spoof the name `Enterprise X IT Staff` and the email address `<helpdesk@enterpriseX.org>` when targeting an organization with a legitimate domain of `enterpriseX.com`.
4. A *lateral attacker* sends their spearphishing email from a compromised, but legitimate, user's email account. From a recipient's perspective, the metadata in a phishing email sent by a *lateral attacker* is indistinguishable from a legitimate email.

Exploit Payload

Once Mallory has gained Alice's trust, she then needs to exploit this trust by inducing Alice to perform some dangerous action. Attackers have typically used three types of exploitation: (i) attachments or URLs that contain malware, (ii) URLs leading to websites that attempt to trick Alice into revealing her credentials, and (iii) out-of-band actions (e.g., tricking a company's CFO into wiring money to a fake, malicious "corporate partner").

Security Model

Threat Model: In this work, we specifically focus on an enterprise *credential spearphishing* threat model, where Mallory tries to fool a targeted enterprise's user (Alice) into revealing her credentials. We assume that the adversary can send arbitrary emails to the victim and can convince the recipient to click on URLs embedded in the adversary's email (leading the victim to a credential phishing website). To impersonate a trusted entity, the attacker may set any of the email header fields to arbitrary values.

In other words, we focus on attacks where Mallory's lure includes masquerading as a trusted entity, her payload is a link to a credential phishing page, and she chooses from any of the last three impersonation models. Because organizations can deploy DKIM/DMARC to mitigate address spoofing (and many large email providers have done so), we exclude address spoofing from our focus. Although recent work has shown that practical attacks could enable an *address spoofer* to bypass these verification frameworks [19, 52], these attacks exploit incorrect implementations and user-interface shortcomings, which email providers have and continue to improve as a result of such research.

Security Goals: First, a detector must produce an extremely low false positive burden, ideally a total daily volume that take at most minutes for an incident response team to process; in our work, we aim for an upper-bound of roughly 10 false alarms per day based on discussions with the security team at LBNL. Second, a detector must detect real spearphishing attacks (true positives).

Data Source	Fields and Information per Log Entry
SMTP logs	Timestamp `From` (header displayed to recipients as the email's sender) `RCPT TO` (list of all recipients; from the SMTP dialog)
NIDS logs	URL visited SMTP log id for the earliest email with this URL Earliest time this URL was visited in HTTP traffic # prior HTTP visits to this URL # prior HTTP visits to any URL with this hostname Hostname (fully qualified domain of this URL) Earliest time any URL with this hostname was visited
LDAP logs	Employee's email address Time of current login Time of subsequent login, if any # total logins by this employee # employees who have logged in from current login's city # prior logins by this employee from current login's city

Table 3.1: Schema for each entry in our data sources. All sensitive information is anonymized before we receive the logs. The NIDS logs contain one entry for each visit to a URL seen in any email. The LDAP logs contain one entry for each login where an employee authenticated from an IP address that he/she has never used in prior successful logins.

Given that current methods for detecting credential spearphishing often rely on users to report an attack, if our approach can detect even a moderate number of true positives or identify undiscovered attacks, while achieving a low false positive rate, then it already serves as a major improvement to the state of detection and mitigation against spearphishing attacks.

3.3 Datasets

Our work draws on the SMTP logs, NIDS logs, and LDAP logs from LBNL; several full-time security staff members maintain these extensive, multi-year logs, as well as a well-documented incident database of successful attacks that we draw upon for our evaluation in Section 3.6. For privacy reasons, before giving us access to the data, staff members at LBNL anonymized all data using the procedure described in each subsection below. Additionally, our anonymized datasets do not contain the contents of email bodies or webpages. Table 3.1 shows the relevant information in these datasets and Table 3.2 summarizes the size and timeframe of our data.

SMTP Logs

The SMTP logs contain anonymized SMTP headers for all inbound and outbound emails during the Mar 1, 2013 – Jan 14, 2017 time period. These logs contain information about all emails sent to and from the organization's employees (including emails between two employees), a total of 372,530,595 emails. The second row of Table 3.1 shows the relevant header information we use for each email from these logs.

The data was anonymized by applying a keyed hash to each sensitive field. Consider a header such as `Alice Good <ali@company.com>`. The "name" of a header is the human name (`Alice Good` in our example); when no human name is present, we treat the email address as the header's "name'. The "address" of a header is the email address: `<ali@company.com>`. Each name and each email address is separately hashed.

NIDS Logs

LBNL has a distributed network monitor (Zeek) that logs all HTTP GET and POST requests that traverse its borders. Each log entry records information about the request, including the full URL.

Additionally, the NIDS maintains a corpus of all URLs seen in the bodies of inbound and outbound emails at LBNL.[1] Each time any URL embedded in an email gets visited as the destination of an HTTP request, the NIDS records information about the request, including the URL that was visited and the entry in the SMTP logs for the email that contained the fetched URL. The NIDS remembers URLs for at least one month after an email's arrival; all HTTP visits to a URL are matched to the earliest email that contained the URL.

We received anonymized logs of all HTTP requests, with a keyed hash applied to each URL and its fully-qualified domain. Additionally, we received anonymized logs that identify each email whose URL was clicked, and anonymized information about the email and the URL, as shown in the third row of Table 3.1.

LDAP Logs

LBNL uses corporate Gmail to manage its employees' emails.[2] Each time an employee successfully logs in, Gmail records the user's corporate email address, the time when the login occurred, and the IP address from which the user authenticated. From these LDAP logs, we received anonymized information about login sessions where (1) the login IP address had never been used by the user during any previous successful login, (2) the user had more than 25 prior logins, and (3) the login IP address did not belong to LBNL's network. The last row of Table 3.1 shows the anonymized data in each entry of the LDAP logs, with the same keyed hash applied to fields such as the email address involved in a login.

[1] Shortened URLs are expanded to their final destination URLs.

[2] Email between two employees also flows through corporate Gmail, which allows our detector to scan "internal" emails for lateral spearphishing attacks.

Time span	Mar 1, 2013 – Jan 14, 2017
Total emails	372,530,595
Unique sender names (names in `From`)	3,415,471
Unique sender addresses (email addresses in `From`)	4,791,624
Emails with clicked URLs	2,032,921
Unique sender names (names in `From`)	246,505
Unique sender addresses (email addresses in `From`)	227,869
# total clicks on embedded URLs	30,011,810
Unique URLs	4,014,412
Unique hostnames	220,932
Logins from new IP address	219,027
# geolocated cities among all new IP addresses	7,937
# of emails sent during sessions where employee logged in from new IP address	2,225,050

Table 3.2: The size of our data across the three log sources. Note that some emails contain multiple URLs, some or all of which may be visited multiple times by multiple recipients; thus, there are more clicked-URLs than emails that contain clicked-URLs.

3.4 Challenge: Diversity of Benign Behavior

Prior work has used machine learning to identify phishing attacks, based on suspicious content in email headers and bodies [29, 114]. While this prior work detects several spearphishing attacks, their optimal false positive rates (FPR) are 1% or higher, which is far too high for our setting: a FPR of 1% would lead to 3.7 million false alarms on our dataset of nearly 370 million.

In this section, we identify several issues that make spearphishing detection a particularly difficult challenge. Specifically, when operating on a real-world volume of millions of emails per week, the diversity of benign behavior produces an untenable number of false positives for detectors that merely look for anomalous header values.

Challenge 1: Senders with Limited Prior History

A natural detection strategy is to compare the headers of the current email under analysis against all historical email headers from the current email's purported sender. For example, consider a *name spoofer* who attempts to phish one of Alice's team members by sending an email with a `From` header of `Alice Good <alice@evil.com>`. An anomaly-based detector could identify

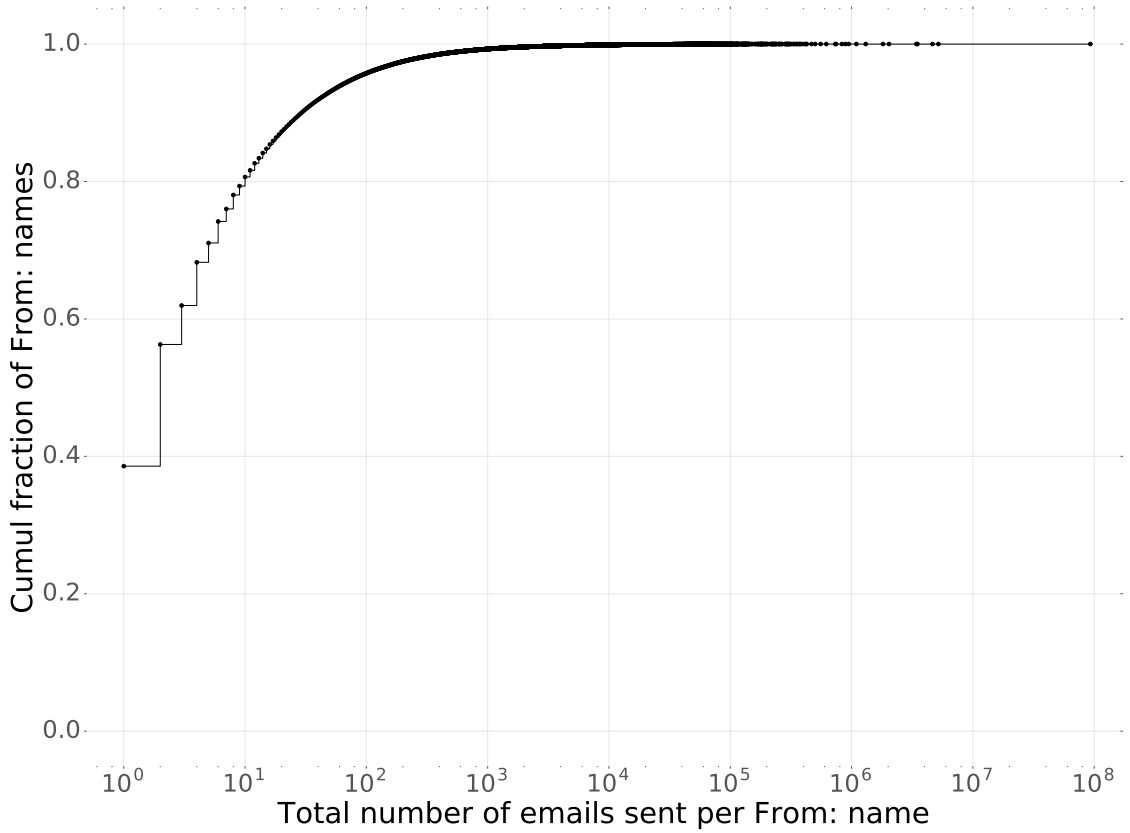

Figure 3.2: Distribution of the number of emails sent per `From` name. Nearly 40% of all `From` names appear in only one email and over 60% of all `From` names appear in three or fewer emails.

this attack by comparing the email's `From` address (`<alice@evil.com>`) against all `From` addresses in prior email with a `From` name of `Alice Good`.

However, this approach will not detect a different spearphishing attack where neither the name nor the address of the `From` header have ever been seen before: `Alice <alice@evil.com>` or `HR Team <hr.enterpriseX@gmail.com>`. In this *previously unseen attacker* setting, there is no prior history to determine whether the `From` address is anomalous.

To address this gap, a detector could flag all emails with a new or previously unknown `From` name (e.g., any email where the `From` name has been seen in two or fewer emails leads to an alert). Unfortunately, this approach generates an overwhelming number of alerts in practice because millions of `From` names are only ever seen in a few emails. Figure 3.2 shows the distribution of the number of emails per `From` name in our dataset. In particular, we find that over 60% of `From` names sent three or fewer emails and over 40% of `From` names sent exactly one email. Thus, even if an organization ran a detector retrospectively to alert on every email with a `From` name that had never been seen before and did not eventually become an active and engaged sender, it would produce over 1.1 million alerts: a false positive rate of less than 1% on our dataset of nearly 370 million emails, but still orders of magnitude more than a practical alert volume. Even

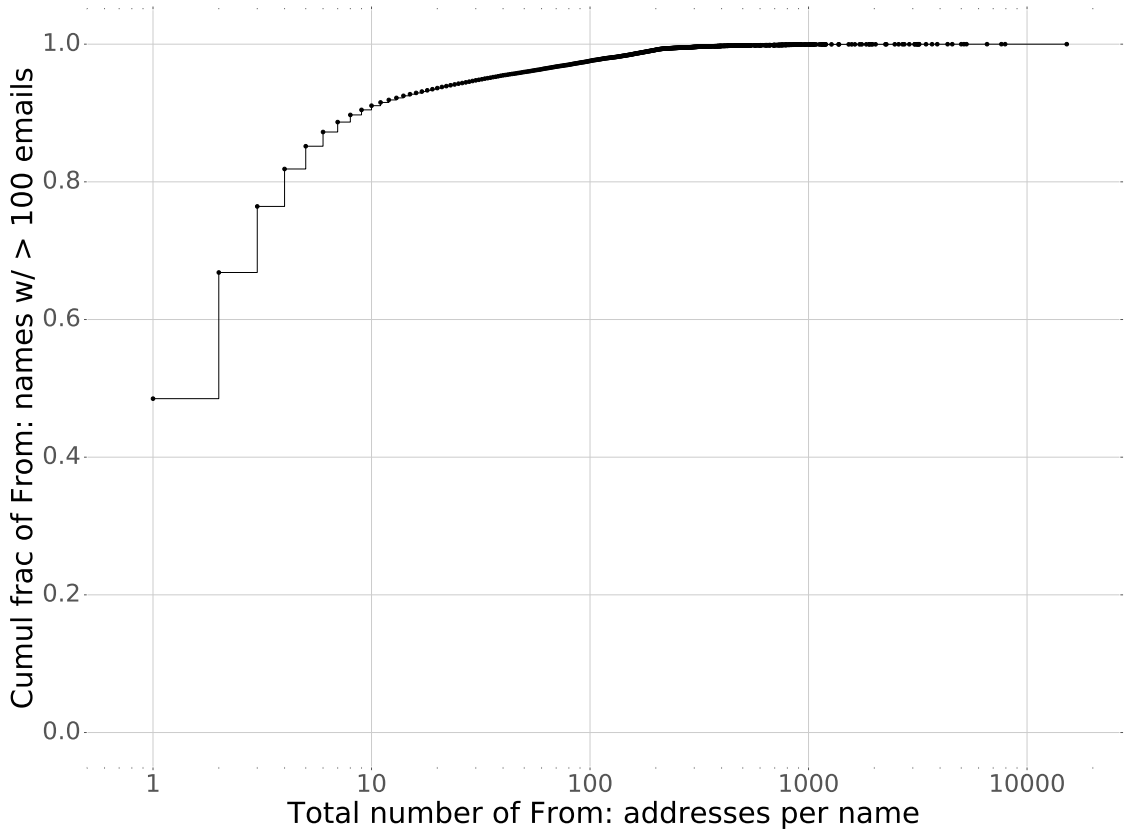

Figure 3.3: Distribution of the total number of `From` addresses per `From` name (who send over 100 emails) across all emails sent by the `From` name. Over half (52%) of these `From` names sent email from two or more `From` addresses (i.e., have at least one new `From` address).

though spam might account for a proportion of these emails with new `From` names, LBNL's security staff investigated a random sample of these emails and found a spectrum of benign behavior: event/conference invitations, mailing list management notices, trial software advertisements, and help support emails. Thus, a detector that only leverages the traditional approach of searching for anomalies in header values faces a stifling range of anomalous but benign behavior.

Challenge 2: Churn in Header Values

Even if we were to give up on detecting attacks that come from previously unseen `From` names or addresses, a detector based on header anomalies still runs into yet another spectrum of diverse, benign behavior. Namely, header values for a sender often change for a variety of benign reasons. To illustrate this, we consider all `From` names that appear in at least 100 emails (our dataset contains 125,172 of them) and assess the frequency at which these names use a new `From` email address when sending email.

Figure 3.3 shows the cumulative distribution of the total number of `From` email addresses per `From` name. From this graph, we see that even `From` names with substantial history (those who have sent over 100 emails) send emails with considerable variability in their header values: 52% of these `From` names send email from more than one `From` email address. We find that 1,347,744 emails contain a new `From` email address which has never been used in any of the `From` name's prior emails. Generating an alert for each of these emails would far exceed our target of 10 alerts per day.

This large number of new email addresses per `From` name stems from a variety of different sources: work vs. personal email addresses for a user, popular human names where each email address represents a different person in real life (e.g., multiple people named `John Smith`), professional society surveys, and functionality-specific email addresses (e.g. `Foo <noreply@foo.com>`, `Foo <help@foo.com>`, `Foo <donate@foo.com>`). While it might be tempting to leverage domain reputation or domain similarity between a new `From` address and the `From` name's prior addresses to filter out false positives, this fails in a number of different cases. For example, consider the case where `Alice` suddenly sends email from a new email address, whose domain is a large email hosting provider; this could either correspond to Alice sending email from her personal email account, or it might represent a *name spoofer* using a Gmail account with a spoofed `From` name.

Given the prevalence of emails with anomalous, yet benign, header values, a practical detector clearly needs to leverage additional signals beyond an email's header values. Some prior academic work has attempted to incorporate stylometry features from an email's body to identify spearphishing attacks [114]; however, as discussed earlier, these systems have false positive rates of 1% or higher, which would lead to millions of false alarms. In the following section, we present a novel approach that leverages a different set of signals based on the underlying nature of spearphishing attacks.

3.5 Detector Design

At a high level, our detector consists of three stages illustrated in Figure 3.4 and described below: a feature extraction stage, a nightly scoring stage, and a real-time alert generation stage. Conceptually, our work introduces two key ideas that enable our detector to uncover a wide range of attacks, while achieving a practical volume of false positives that is over 200 times lower than prior work. First, our detector extracts two sets of reputation-based features that independently target the two key stages of a spearphishing attack identified in our attack taxonomy. Second, we introduce a novel, unsupervised anomaly detection technique that enables our detector to automatically rank a set of unlabeled events and select the most suspicious events for the security team to review. We first discuss each of these elements and then show how to combine them for a real-time detector.

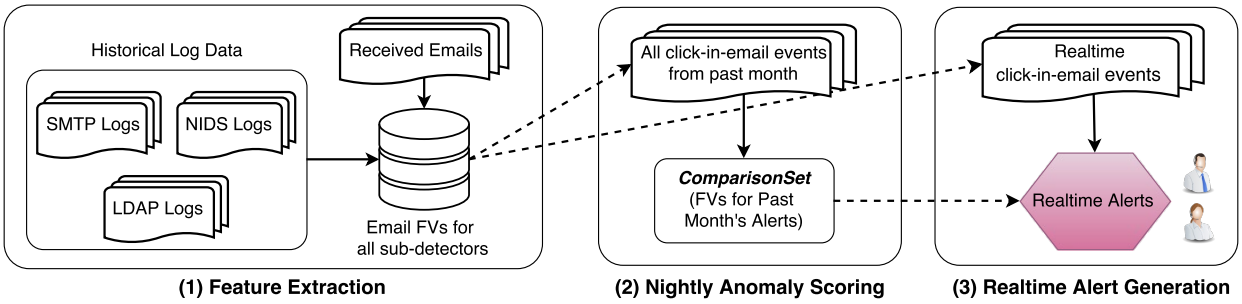

Figure 3.4: Overview of our real-time detector, which leverages output from a nightly batch job during its real-time analysis, as described in Section 3.5. As emails arrive, our detector leverages historical logs to extract and save three feature vectors (one feature vector per impersonation model) for each URL in the email. Using the network traffic logs, our detector logs all clicks on URLs embedded in emails. Each night, our detector runs our anomaly scoring algorithm on the feature vectors (FVs) over a sliding window from the past month's clicked URLs and stores a *ComparisonSet* of the month's most suspicious FVs for each impersonation model. Observing real-time network traffic, our detector sees new click-in-email events, compares the real-time event's feature vector for each impersonation model against the *ComparisonSet*, and generates an alert for the security team if needed.

Features per Attack Stage

Fundamentally, spearphishing attacks aim to trick their recipients into performing a dangerous action described in the email. If the attacker fails to persuade the victim into taking the action, the attack fails. For credential spearphishing, the dangerous action is clicking on a link in an email that leads the victim to a credential phishing website.[3] Thus, we analyze every email that contains a URL that a user clicked on; we call this clicked link a *click-in-email* event.

As discussed in our taxonomy (Section 3.2), spearphishing attacks consist of two necessary stages: the lure stage, where the attackers persuade the victim to trust them, and the exploit stage, where the victim performs a dangerous action for the attackers. This insight leads to the first core idea in our approach: we craft two sets of features to target both of these stages of a spearphishing attack. Prior work has often used features that capture only the lure or the exploit; our insight is that we can do significantly better by using both types of features.

Accordingly, we have two classes of features: *domain reputation* features, and *sender reputation* features. In order to steal the victim's credentials, the attacker must link to a site under her control. Because spearphishing attacks are so tightly targeted, visits to this malicious website will presumably be rare among the historical network traffic from the organization's employees. Therefore, for each click-in-email event, the domain reputation features characterize the likelihood that an employee would visit that URL, based on its (fully qualified) domain. The sender reputation features characterize whether the sender of that email falls under one of the impersonation models

[3]While an adversary could attempt to spearphish an employee's credentials by fooling them into including the credentials in an email response, this attack variant is likely more difficult to successfully execute given employee awareness from security training and education. Based on their multi-year incident database, LBNL has not observed such attacks succeed in practice.

outlined in our taxonomy. Effectively, the sender reputation features capture elements of the lure stage (by recognizing different types of spoofing that the attacker might use to gain the victim's trust), and the domain reputation features capture characteristics of the exploit stage.

Because the sender reputation features differ for each impersonation model, our detector actually consists of three sub-detectors, one for each impersonation model. As discussed below, if any of the sub-detectors flags an email as spearphishing, the detector treats it as an attack and generates an alert for the security team.

Features

Given a click-in-email event to classify, each sub-detector computes a feature vector containing four scalar values, two for domain reputation and two for sender reputation; Appendix A.1 contains a summary table of these features, which we discuss below. As we show later (Section 3.6), these compact feature vectors suffice to detect a wide-range of attacks while achieving a practical volume of false positives.

Domain Reputation Features

All sub-detectors use the same two features to characterize the reputation of a link that the user clicked on. Intuitively, if few employees from the enterprise have visited URLs from the link's domain, then we would like to treat a visit to the email's link as suspicious. Additionally, if employees have never visited URLs from a domain until very recently, then we would also like to treat visits to the domain's URLs as risky. Based on these two ideas, the first feature counts the number of prior visits to any URL with the same fully qualified domain name (FQDN) as the clicked URL; this is a global count across all employees' visits, from the NIDS logs. The second feature counts the number of days between the first visit by any employee to a URL on the clicked link's FQDN and the time when the clicked link's email initially arrived at LBNL.

We chose to characterize a clicked link's reputation in terms of its FQDN, rather than the full URL, because over half of the clicked URLs in our dataset had never been visited prior to a click-in-email event. Consequently, operating at the granularity of the full URL would render the URL reputation features ineffective because the majority of URLs would have the lowest possible feature values (i.e., never been visited prior to the email recipient). Additionally, using a coarser granularity such as the URL's registered domain name or its effective second-level domain could allow attackers to acquire high reputation attack URLs by hosting their phishing webpages on popular hosting sites (e.g., attacker.blogspot.com). By defining a URL's reputation in terms of its FQDN, we mitigate this risk.

Sender Reputation Features

Name Spoofer: As discussed earlier (Section 3.2), in this attacker model Mallory masquerades as a trusted entity by spoofing the name in the `From` header, but she does not spoof the name's true email address (e.g., to avoid a DMARC warning in the recipient's mail client). Because the trusted

user that Mallory impersonates does not send email from the spearphishing email's chosen address, the email will have a `From` address that does not match any of the historical email addresses for its `From` name. Therefore, the first sender reputation feature counts the number of previous days where that had an email whose `From` header contains the same name and address as the email being scored. This captures the intuition that our detector should treat emails from a user that contain a new or rarely-used email address with greater suspicion.

Additionally, in this attacker model, the adversary spoofs the `From` name because the name corresponds to someone known and trusted. If that name did not correspond to someone trustworthy or authoritative, there would be no point in spoofing it; or the attack would manifest itself under our *previously unseen attacker* threat model. Thus, the second sender reputation feature for emails under this impersonation model reflects the trustworthiness of the name in its `From` header. We measure the trustworthiness of a name by counting the total number of weeks where an email's name sent at least one email for every weekday of the week. Intuitively, `From` names that frequently and consistently send emails will be perceived as familiar and trustworthy.

Previously Unseen Attacker: Under this attacker model, Mallory chooses a name and email address that resemble a known or authoritative entity, but where the name and email address do not exactly match any existing entity's values (e.g., `IT Support Team <helpdesk@company.net>`). If the name or address did exactly match an existing entity, the attack would instead fall under the *name spoofer* or *address spoofer* threat model.

Compared to name-spoofing attacks, these attacks are more difficult to detect because we have no prior history to compare against; indeed, prior work does not attempt to detect attacks from this threat model. To deal with this obstacle, we rely on an assumption that the attacker will seek to avoid detection, and thus the spoofed identity will be infrequently used; each time Mallory uses the spoofed identity, she runs the risk that the employee she interacts with might recognize that Mallory has forged the name or email address and report it to the organization's security team. Accordingly, we use two features: the number of prior days that the `From` name has sent email, and the number of prior days that the `From` address has sent emails.

Lateral Attacker: Our detector also aims to catch spearphishing emails sent from a compromised user's accounts (without using any spoofing). To detect this powerful class of attackers, we leverage the LDAP logs provided by Gmail's corporate email services (Section 3.3). When a recipient clicks on a link in an email, if the email was sent by an employee, we check the LDAP logs to see if the email was sent during a login session where the sender-employee logged-in using an IP address that the sender-employee has never used before. If so, this sub-detector computes the geolocated city of the session's IP address, say city C. It then extracts two features: the number of distinct employees that have logged in from city C, and the number of previous logins where this sender-employee logged in from an IP address that geolocated to city C.

Content-Based Features: As discussed in Section 3.3, for privacy reasons we do not have access to either the bodies of emails or the contents of a clicked URL's webpage. If desired, enterprises could augment our sender reputation features with additional features from the raw content in the

email message or website (e.g., NLP features that characterize whether the email message relates to accounts and credentials or reflects particular sentiments such as urgency).

Limitations of Standard Detection Techniques

Once our detector has extracted features for each click-in-email event, it needs to decide which ones should trigger an alert for the security team. We first discuss three natural, but ultimately ineffective, approaches for determining which events to alert on. Then, in the following subsection, we present a new technique, DAS, that our detector uses to overcome the limitations of these canonical approaches.

Manual Thresholds: The simplest classification approach would be to manually select a threshold for each feature, and generate an alert if all feature values are below the threshold. One might use domain knowledge of each feature to guess a threshold for each feature dimension: e.g., spearphishing attacks will use URLs whose domain has fewer than five visits or was first visited less than five days ago. Unfortunately, this approach is inherently arbitrary since we do not know the true distribution of feature values for spearphishing attacks. Thus, this ad hoc approach can easily miss attacks, and does not provide a selection criteria that generalizes across different enterprises.

Supervised Learning: A large body of literature on attack detection, from spam classification to prior spearphishing work, draws heavily on supervised machine learning algorithms. However, those methods face critical shortcomings in our setting.

To accurately classify new events, supervised learning techniques require a labeled training dataset that reflects the range of possible malicious and benign feature values. Unfortunately, in our context, we lack a sufficiently large set of labeled attacks to train a functional supervised learning model; spearphishing attacks occur at a low rate and are extremely difficult to detect when they do occur.

Additionally, our setting exhibits extreme class imbalance: because of the scarcity of spearphishing attack data, the training set will have vastly more benign instances than malicious instances. Supervised techniques often need a relatively balanced dataset; classifiers trained on highly imbalanced data often learn to always predict the majority class (i.e., classify everything as benign), pathologically overfit to accidental characteristics of the minority class, or generate too lax of a decision boundary and generate prohibitively high numbers of false positives [47]. While the machine learning community has explored a number of techniques for addressing imbalanced training data [18, 47], such as undersampling the over-represented class or synthetically generating samples for the under-represented class, these techniques do not scale to imbalances on the order of millions to one.

Standard Anomaly Detection: Alternatively, a detector could draw upon unsupervised or semi-supervised anomaly detection techniques. While a number of such techniques exist, including density estimation techniques such as Gaussian Mixture Models (GMMs) [17] and clustering or distance-based techniques such as k-nearest-neighbor (kNN) [63], these classical techniques suffer from three limitations.

Algorithm 1 Scoring and Alert Selection in DAS

```
Score(E, L):
for each event X in L do:
    if E is more suspicious than X in every dimension:
        Increment E's score by one

AlertGen(L (a list of events), N):
for each event E in L do:
    Score(E, L)
Sort L by each event's score
return the N events from L with the highest scores
```

First, in a number of security settings, scalar features often have a directionality to their values; indeed, all of our features have this property. For example, the fewer visits a domain has, the more suspicious it is; an unusually small number of visits is grounds for suspicion, but an unusually large number is not. Standard anomaly detection techniques do not incorporate notions of asymmetry or directionality into their computations. For example, density-based anomaly detection techniques such as kernel density estimation (KDE) and GMMs fit a probability distribution to the data and alert on the lowest-probability events. Events that have statistically extreme, but benign, feature values will have a very low probability of occurring, triggering a large number of useless alerts.

Second, standard anomaly detection techniques often treat an event as anomalous even if only one or a few of the event's features are statistically anomalous. However, in our setting, we expect that attacks will be anomalous and suspicious in all feature dimensions. Consequently, in our setting, classical techniques will generate many spurious alerts for events that are only anomalous in a few dimensions. As we show in Section 3.6, this causes classical techniques to miss the vast majority of spearphishing attacks in our dataset because they exhaust their alert budget with emails that have benign feature values in all-but-one dimension.

Third, classical techniques are parametric: they either assume the data comes from a particular underlying distribution, or they contain a number of parameters that must be correctly set by their deployer in order for the technique to obtain acceptable performance. GMMs assume the data comes from a mixture of Gaussian distributions, KDE has a bandwidth parameter that requires tuning by the deployer, and kNN needs the deployer to select a value of k (the number of nearest neighbors/most similar events, which the algorithm will use to compute an event's anomaly score). These requirements are problematic for spearphishing detection since we do not know the true distribution of attack and benign emails (e.g., the true distributions might not be Gaussian) and we do not have a sound way to select the parameters.

Directed Anomaly Scoring (DAS)

Given the limitations of traditional detection techniques, we introduce a simple and general technique for automatically selecting the most suspicious events from an unlabeled dataset. We call our

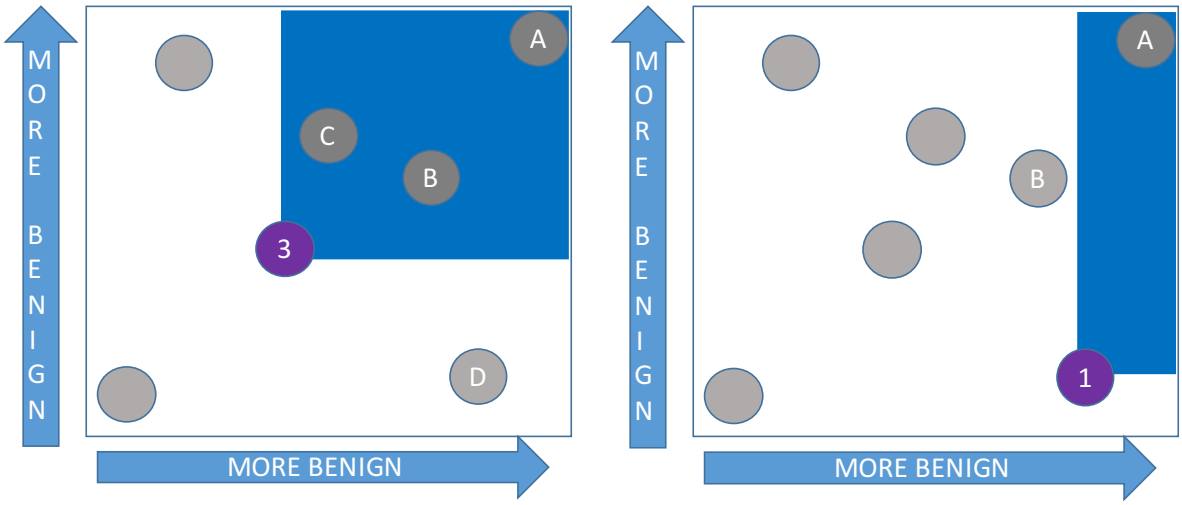

Figure 3.5: Examples of DAS scoring for events in a 2 dimensional feature space. X-values to the right and Y-values toward the top are more benign (thus, values toward the bottom and left are more suspicious). Each circle represents an example event. The number in each circle is the DAS score for the event. For example, in the diagram on the right, the purple event only receives a score of 1. Although the purple event has a more suspicious feature value in the Y dimension than event B, it is more benign in the X dimension. Thus, event B does **not** cause the purple event's score to increment.

technique Directed Anomaly Scoring (DAS). At a high level, DAS ranks all events by comparing how suspicious each event is relative to all other events. Once all events have been ranked, DAS simply selects the N most suspicious (highest-ranked) events, where N is the security team's alert budget.

Algorithm 1 shows the procedure for scoring and generating alerts with DAS. Concretely, DAS first assigns an anomaly score for each event, E, by computing the total number of other events where E's feature vector is at least as suspicious as the other event in every feature dimension. Thus, E's score counts how many events it is at least as suspicious as; events with higher scores are more suspicious than ones with lower scores. Figure 3.5 presents a few visual examples of computing DAS scores. After scoring every event, our algorithm simply sorts all events by their scores and outputs the N highest-scoring events.

Formally, we identify each event with its feature vector $E \in \mathbb{R}^d$. We consider event E to be at least as suspicious as event E', written $E \succcurlyeq E'$, if $E_i \leq E'_i$ for all $i = 1, 2, \ldots, d$. Then, the score of event E is the cardinality of the set $\{E' : E \succcurlyeq E'\}$. (For simplicity, we assume that smaller feature values are more suspicious, in every dimension; for dimensions where the reverse is true, we replace the comparator $\leq$ with $\geq$. Appendix A.1 summarizes the comparators we use for each feature.)

DAS is well-suited for a range of security detection problems where attacks can be characterized by a combination of numerical and boolean features, such as our spearphishing use case. As we show in Section 3.6, DAS achieves orders-of-magnitude better results than classical anomaly detection techniques because it leverages domain knowledge about which regions of the feature

space are most suspicious; in particular, it overcomes all three limitations of classical techniques discussed in Section 3.5.

Real-time Detection Architecture

We now synthesize the ideas discussed in previous subsections to provide an end-to-end overview of how we leverage our features and DAS to generate alerts (illustrated in Figure 3.4). Our detector has access to the enterprise's email and LDAP data, real-time network traffic (e.g., via a NIDS like Zeek), and an alert budget β for each sub-detector, which specifies the daily volume of alerts that the security team deems acceptable. As each email arrives, for each URL in the email, our detector extracts the feature vectors for that URL and saves them in a table indexed by the URL; i.e., for each URL in each email, this indexed table stores a set of feature vectors, with one feature vector for each sub-detector. Monitoring real-time HTTP requests seen by the enterprise's NIDS, our detector checks each URL request against the indexed table to see if it corresponds to a "click-in-email" event (a visit to a URL contained within an email). If our detector observes a click-in-email event, it adds that URL and its feature vectors to a list of candidate alerts. Finally, our detector gathers a set of click-in-email events and uses the DAS algorithm to rank these events and determine which ones to alert on.

This approach would work fine for a batch algorithm, e.g., running the DAS algorithm once a month on the past thirty day's candidate alerts. However, a batch detector will suffer from a delay between when an attack occurred and the next time an organization runs the batch detection algorithm.

Therefore, we use a more sophisticated algorithm that includes a batch computation step, yet operates in real time. Each night, our detector collects all the click-in-email events for the past month and computes their associated feature vectors. For each sub-detector, our system ranks these events using DAS, selects the $30 \times \beta$ most suspicious events, and saves them in a set that we call the *ComparisonSet*. In real-time, when our detector observes a click-in-email event from the NIDS, it fetches the event's feature vectors for each impersonation model. Our detector then computes whether any of the current click's feature vectors are at least as suspicious as any of the feature vectors in the *ComparisonSet* for each respective impersonation model.[4] If so, our detector generates an alert for the security team.

Intuitively, this approach produces an alert if the event would have been selected by DAS on any day in the past month; or, more precisely, if it is among the 30β most suspicious events in the past month. Our evaluation (Section 3.6) shows that this real-time approach can safely detect the same attacks as the batch scoring procedure. On some days our real-time approach might generate more alerts than the target budget if a day has a burst of particularly suspicious click-in-email events; however, we show in the next section that this occurs infrequently in practice.

[4]This is equivalent to running DAS to score the current feature vector against the *ComparisonSet* and checking whether it gives the current feature vector a score of at least 1.

Alert Classification	*Name spoofer*	*Previously unseen attacker*	*Lateral attacker*	Total Count
Spearphish: *known* + successful attack	2	2	2	6 / 7
Spearphish: **unknown** + successful attack	1	1	0	2 / 2
Spearphish: failed attack	3	6	0	9 / 10
Total Spearphish Detected	6	9	2	17 / 19

Table 3.3: Summary of our real-time detection results for emails in our test window from Sep 1, 2013 - Jan 14, 2017 (1,232 days). Rows represent the type/classification of an alert following analysis by security staff members at LBNL. Columns 2–4 show alerts broken down per attacker model (Section 3.5). Column 5 shows the total number of spearphishing campaigns identified by our real-time detector in the numerator and the total number of spearphishing campaigns in the denominator. Out of 19 spearphishing attacks, our detector failed to detect 2 attacks (one that successfully stole an employee's credentials and one that did not); both of these missed attacks fall under the *previously unseen attacker* threat model, where neither the username nor the email address matched an existing entity.

3.6 Evaluation and Analysis

We evaluated our real-time detector on our dataset of 370 million emails from LBNL, measuring its detection performance (true positives), the time burden (false positives) it imposes on an enterprise's security staff, and how it performs relative to standard anomaly detection techniques that use the same set of features.

For each click-in-email event, we computed its reputation features using log data from a sliding window over the six months prior to the click event. To bootstrap this process, we use the first six months of our dataset as a burn-in period and do not generate alerts for any emails in that period. Later in Section 3.7, we explore the impact of using a smaller window of historical data to compute feature values.

We configured our detector with a daily budget of 10 alerts per day. LBNL's security team specified 10 alerts per day as a very tolerable number since their team consists of several analysts who routinely process a few hundred alerts each day. To divide this budget among each of our three sub-detectors, we allocated 4 alerts per day for each of the *name spoofer* and *previously unseen attacker* sub-detectors and 2 alerts per day for our *lateral attacker* sub-detector; since lateral spearphishing requires the use of a compromised account, we expect it to occur less often than spoofing-based spearphishing.

Detection Results: True Positives

Because spearphishing attacks occur infrequently and often go undetected, developing ground truth and measuring true positives is a hard problem. For our evaluation, we draw upon LBNL's incident database, which contains 7 known successful spearphishing attacks; this includes 1 spearphishing exercise by an external security firm (conducted independently of our work) that successfully stole employee credentials. Additionally, members of LBNL's security team manually investigated and labeled 15,521 alerts. We generated these alerts from a combination of running (1) an older version of our detector that used manually-chosen thresholds instead of the DAS algorithm; and (2) a batched version of our anomaly scoring detector, which ran the full DAS scoring procedure over

the click-in-email events in our evaluation window (Sep 2013 onward) and selected the highest scoring alerts within the cumulative budget for that timeframe.

From this procedure, we identified a total of 19 spearphishing campaigns: 9 which succeeded in stealing an employee's credentials and 10 where the employee clicked on the spearphishing link, but upon arriving at the phishing landing page, did not enter their credentials.[5] We did not augment this dataset with simulated or injected attacks (e.g., from public blogposts) because the true distribution of feature values for spearphishing attacks is unknown. Even for specific public examples, without actual historical log data one can only speculate on what the values of our reputation features should be.

To evaluate our true positive rates, we ran our real-time detector (Section 3.5) on each attack date, with a budget of 10 alerts per day. We then computed whether any of our detector's alerts flagged each respective attack campaign on those dates. Table 3.3 summarizes our evaluation results. Overall, our real-time detector successfully identified 17 out of 19 spearphishing campaigns: an 89% true positive rate. Among the attacks that our detector correctly identified, manual analysis by staff members at LBNL indicated that our sub-detectors aptly detected spearphish that fell under each of their respective threat models (outlined in Section 3.2).

Of the successful spearphishing campaigns originally listed in LBNL's incident database, our detector correctly produced alerts for 6 out of the 7 known attacks.[6] The missed attack used a now-deprecated feature from Dropbox [25] that allowed users to host static HTML pages under one of Dropbox's primary hostnames, which is both outside of LBNL's NIDS visibility because of HTTPS and inherits Dropbox's high reputation. This represents a limitation of our detector: if an attacker can successfully host the malicious phishing page on a high-reputation site or outside of the network monitor's visibility, then we will likely fail to detect it. However, Dropbox and many other major file sharing sites (e.g., Google Drive) have dropped these website-hosting features due to a number of security concerns, such as facilitating phishing. Ironically, in the specific case of Dropbox, industry reports mention a large increase in phishing attacks targeted against Dropbox users, where the phishing attack would itself be hosted via Dropbox's website hosting feature, and thus appear to victims under Dropbox's real hostname [56]. Furthermore, although our detector missed one successful attack, it identified 2 previously undiscovered attacks that successfully stole an employee's credentials.

False Positives and Burden of Alerts

At a daily budget of 10 alerts per day, our detector achieved an average false positive rate of 0.004% (the median number of emails per day is 263,086). However, as discussed earlier in Section 3.5, our real-time detector might not produce exactly 10 alerts per day; some days might have a burst of particularly suspicious emails, while other days might not have any unusual activity at all. To evaluate the actual daily alert load, we ran our real-time detector on one-hundred randomly selected days in our dataset and computed the total number of alerts it generated on each day,

[5] A campaign is identified by a unique triplet of ⟨the attack URL, email subject, and email's `From` header⟩.

[6] LBNL's incident database catalogues successful attacks, but not ones that failed.

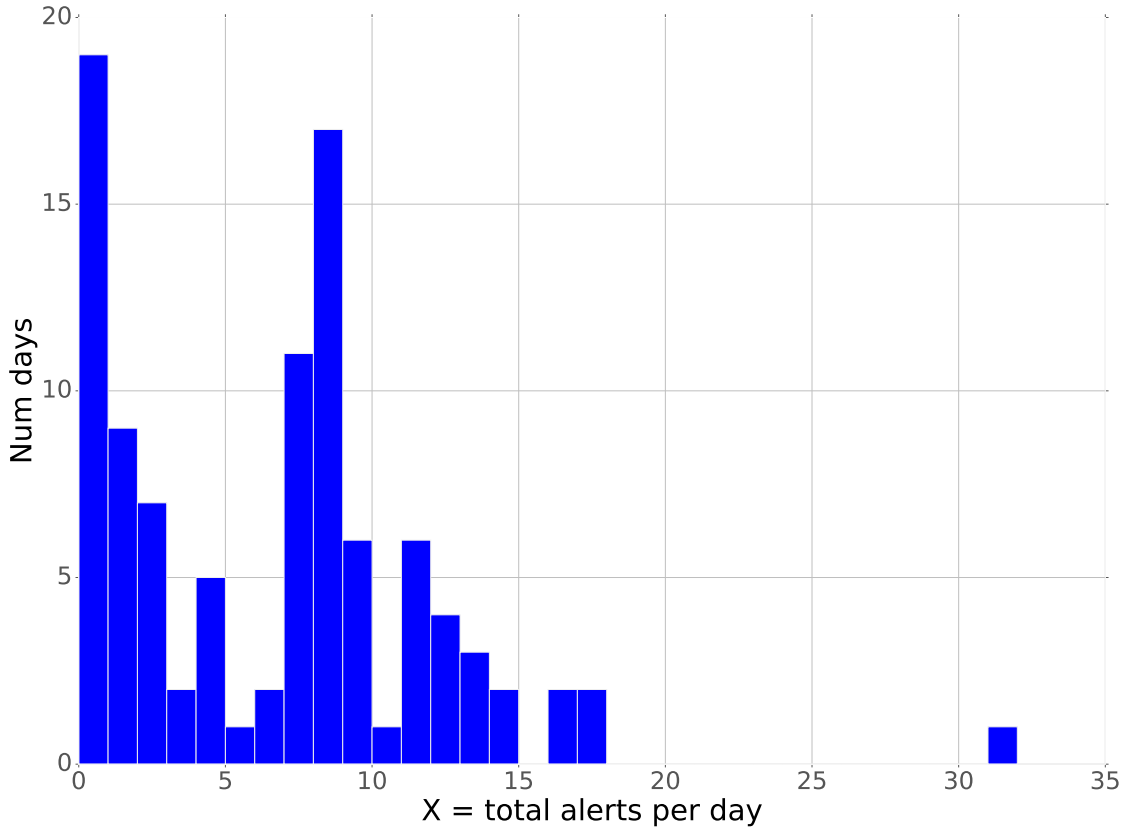

Figure 3.6: Histogram of the total number of daily alerts generated by our real-time detector (cumulative across all three sub-detectors) on 100 randomly sampled days. The median is 7 alerts/day.

shown in Figure 3.6. From this histogram, we see that while our detector occasionally generates bursts over our target budget, on the vast majority of days (80%) it generates 10 or fewer alerts per day; on nearly 20% of days, it generates no alerts.

During their manual investigation of the 15,521 alerts created during our ground truth labeling process, LBNL's security staff tracked how long it took them to investigate these alerts. Surprisingly, LBNL's security staff reported that a single analyst could process an entire month's worth of alerts in under 15 minutes; and thus, on average, take under one minute to analyze one day's worth of alerts.

This rapid processing time arises because the analysts were able to develop a two-pass workflow that enabled them to quickly discard over 98% of the alerts during a fast triaging pass, at a rate of 2 seconds per alert; and then follow up with a more in-depth analysis pass for the remaining 2% of alerts, at a rate of 30 seconds per alert (e.g., analyzing detailed HTTP logs and examining the full email headers). The first pass is so fast because, for the vast majority of our detector's alerts, an analyst could quickly determine if an email constituted a plausible spearphishing threat by inspecting the Subject line, `From` line, and clicked URL of the email. For over 98% of our alerts, this trio of information indicated that the email was highly unlikely to contain a credential

Algorithm	Detected	Daily Budget
kNN	3/19	10
	17/19	2,455
GMM	4/19	10
	17/19	147
KDE	4/19	10
	17/19	91
DAS (Section 3.5)	17/19	10

Table 3.4: Comparing classical anomaly detection techniques to DAS, on the same dataset and features. For each of the standard anomaly detection algorithms, the first row shows the number of attacks detected under the same daily budget that we used for DAS in our real-time detector; the second row shows what the classical technique's budget would need to be to detect all 17 attacks that our real-time detector, leveraging DAS, identified on a budget of 10 alerts per day.

spearphishing attack. For example, emails with subjects such as "Never Lose Your Keys, Wallet, or Purse Again!" and "ATTN: Your Stomach Issues FINALLY Explained. See Video Here" are surely not spearphishing attacks.

While the more time-intensive 2% of alerts contained mostly false positives (i.e., not spearphishing), the analysts found two interesting classes of alerts. First, in addition to detecting spearphishing attacks, our detector identified 41 emails from "regular" phishing campaigns. The analysts distinguished between regular phishing and spearphishing by checking whether the email and HTTP response from the clicked URL contained content that was specifically targeted at LBNL. Second, ironically, our detector generated 40 alerts where the person who clicked on the link in the email was not one of the email's recipients, but rather a member of LBNL's security staff. These clicks were part of routine investigations conducted by LBNL's security staff; for example, in response to a user reporting a suspicious email.

Anomaly Detection Comparisons

In Section 3.5 we introduced DAS, a simple new technique for anomaly detection on unlabeled data. Now, we evaluate the effectiveness of DAS compared to traditional unsupervised anomaly detection techniques.

We tested three common anomaly detection techniques from the machine learning literature: Kernel Density Estimation (KDE), Gaussian Mixture Models (GMM), and k-Nearest Neighbors (kNN) [17]. To compare the real-time detection performance of each of these classical techniques against DAS's real-time performance, we ran each of these classical techniques using the same training and evaluation procedures we used for our real-time detector's evaluation. Specifically, given the date of each of the 19 attacks and its impersonation model, we extracted the same exact feature values for all click-in-email events that occurred within a thirty day window ending on the attack date; the thirty day window mirrors the timeframe we used to construct our detector's *ComparisonSet*. We then normalized these feature values and ran each of the three classical anomaly

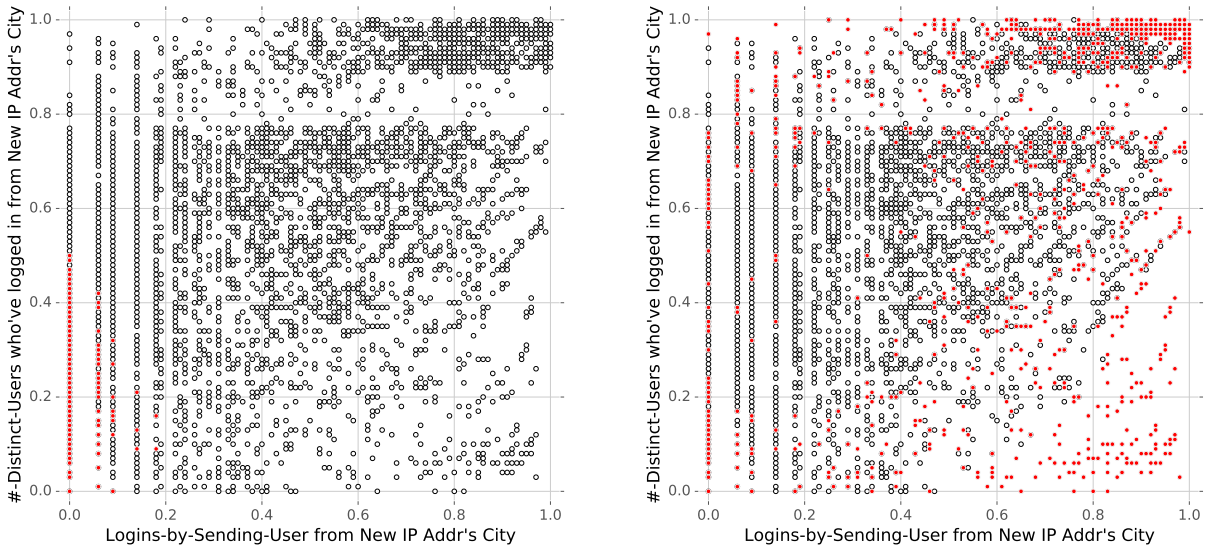

Figure 3.7: Both plots show the sender reputation feature values (scaled between [0, 1]) of a random sample of 10,000 *lateral attacker* click-in-email events. Filled red points denote events that generated alerts within the daily budget by DAS (left-hand figure) and KDE (right-hand figure).

detection techniques on this set of click-in-email events for each attack date. For quantitative comparisons, we computed (1) the number of attacks that would have been detected by each classical technique if it used the same budget that our real-time detector used and (2) the daily budget the classical technique would need to detect all of the attacks that our DAS-driven detector identified.

Like other machine learning methods, these classical algorithms require the user to set various hyperparameters that affect the algorithm's performance. For our evaluation, we tested each classical technique under a range of different hyperparameter values and report the results for whichever hyperparameters gave the best results (i.e., comparing DAS against the best-case version of these classical techniques).

Table 3.4 summarizes the results of this experiment. Under the same daily budget of 10 alerts per day, all three traditional techniques detected fewer than 25% of the attacks found by DAS. Moreover, in order for the best performing classical technique (KDE) to detect as many attacks as DAS, it would need a daily budget nearly an order of magnitude larger than ours.

To illustrate why standard unsupervised techniques perform so poorly, the two plots in Figure 3.7 show the sender reputation features for a random sample of 10,000 *lateral attacker* click-in-email events. The left plot shows the feature values for the actual alerts our DAS-driven detector generated (in red), while the right plot shows the feature values for the alerts selected by KDE using the same budget as our detector. KDE selects a mass of points in the upper-right corner, which illustrates one of limitations of standard techniques discussed in Section 3.5: they do not take into account the directionality of feature values. Because extremely large feature values occur infrequently, KDE ranks those events as highly anomalous, even though they correspond to benign login sessions where the user happened to login from a new IP address in a residential city nearby LBNL.

Second, KDE selects a group of events in the bottom-right corner, which correspond to login sessions where an employee logged in from a city that they have frequently authenticated from in the past, but where few other employees have logged in from. KDE's selection of these benign logins illustrates another limitation of standard techniques: they often select events that are anomalous in just one dimension, without taking into account our domain knowledge that an attack will be anomalous in all dimensions. Even though the bottom-right corner represents employee logins where few other employees have logged in from the same city, they are not suspicious because they correspond to a common legitimate login process: benign logins by remote employees who live and work from cities far from LBNL's main campus. Thus, DAS can significantly outperform standard unsupervised anomaly detection techniques because it allows us to incorporate domain knowledge of the features into DAS's decision making.

3.7 Discussion and Limitations

Detection systems operate in adversarial environments. While our approach can detect both known and previously undiscovered spearphishing attacks, there are limitations and evasion strategies that adversaries might pursue.

Limited Visibility: Our detection strategy hinges on identifying if an email's recipient engaged in a potentially dangerous action. In the case of credential spearphishing, LBNL's network traffic logs allowed us to infer this behavior. However, our approach has two limitations: first, email and network activity conducted outside of LBNL's network borders will not get recorded in the NIDS logs. Second, LBNL made a conscious decision not to man-in-the-middle traffic served over HTTPS; thus, we will miss attacks where the email links to an HTTPS website. Both of these are typical challenges that network-level monitoring faces in practice. An organization could alleviate both of these problems by deploying endpoint monitoring agents on employee machines that record all network activity on each user's device. Alternatively, a site could instrument and/or rewrite all URLs embedded within emails to redirect the user through an enterprise's monitoring site, allowing the organization precisely identify when one of its users clicks on embedded URLs; this approach would resolve any visibility limitations for users operating outside of the enterprise's network borders. To handle challenges posed by HTTPS traffic to and from the enterprise's network, a detector could leverage SNI [125] to develop its domain reputation features for HTTPS traffic and identify when users visit potentially dangerous HTTPS domains.

In addition to limited network visibility, our detector might miss attacks if a spearphishing email came from a compromised personal email account. Since our detector relies on access to a user's prior login information to detect lateral spearphishing attacks, it will not have the necessary data to compute the features for this sub-detector, since organizations presumably lack visibility into their users' personal account activity. To defend against this genre of lateral spearphishing, one could leverage alternative sender reputation features, such as ones based on stylometry [29, 114].

False Negatives and Evasion Strategies: Our detector attempts to adhere to an upper-bound on the number of alerts it generates. As a result, it might miss some attacks if a number of successful

spearphishing campaigns occur on a given day; in effect, the clicks on URLs from the campaigns earlier in the day will mask campaigns that occur later on. To overcome this problem, the security staff could increase the detector's alert budget on days with many attack alerts.

Aside from trying to mask one attack campaign with another, an adversary could attempt to escape detection by crafting an email whose domain or sender reputation features are high. Attackers could boost their link's domain reputation by compromising a frequently visited website and using it to host the credential spearphishing website. This strategy incurs greater costs to execute than modern-day attacks (where an adversary can simply setup her own cheap phishing webpage), and it is unclear whether such an attack would succeed if the compromised site does not normally ask for the employee's corporate credentials. For example, even if an adversary compromises a popular video website (e.g., netflix.com), many users might find it unusual for that popular domain to suddenly start asking for the user's enterprise credentials.

Alternatively, an attacker could attempt to inflate the sender reputation features of their adversarial email before using it in an attack. For instance, to prepare a malicious email address for a name spoofing attack, an adversary could start sending emails with the malicious email address and spoofed `From` name for several days before sending a spearphishing email to the targeted recipient. However, the more frequently an attacker uses this spoofed email address, the more the adversary risks someone detecting the use of a spoofed name; thus this evasion strategy does incur a cost and risk to the attacker.

To make evasion even more costly, future work could explore methods to make DAS more robust. In particular, rather than treating an event E as more suspicious than another event X only if E is more suspicious than X in every dimension, a more robust scoring algorithm might treat E as more suspicious if it is more suspicious than X in at least k dimensions.

Prior History for Feature Extraction: For each click-in-email event, our detector leveraged 6 months of prior log data in order to compute meaningful reputation features. LBNL stores several years worth of logs, so our detector could easily use this amount of prior history in practice. However, with less historical data, the quality of our detector might degrade; e.g., in the degenerate case with no prior history, all `From` names and addresses will appear as suspicious new entities. To assess how much history our detector needs, we re-ran our evaluation experiments (Section 3.6) with only 3 months of history and with 1 month of history for our detector's feature computation. A 3-month historical window sufficed to detect the same attacks as our 6-month real-time detector, and the median number of alerts per day remained the same (7 per day). However, a detector with only 1 month of history failed to detect one of the attacks and generated a median of 18 alerts per day. With just one month of prior data, too many click-in-email events have the smallest possible feature values; this causes our detector to select entire batches of them because they share the same DAS score.

Extending to Preventative Protection: One could extend our real-time detector to operate in a preventative fashion. As emails arrived, our detector could compute each email's feature values and then check each URL in the email to see whether or not it would generate an alert if the URL were clicked at that moment. If so, our system could rewrite the email's URL (before delivering the email to its recipient) to point to an interstitial warning page set up by the enterprise's security

team. Our computations show that if we used our real-time detector with a budget of 10 alerts/day, an employee would encounter a median of 2 interstitial pages over the nearly 4-year time span of our evaluation data (Appendix A.2). Given this low burden, future work could explore how to design effective warning mechanisms as part of a preventative defense.

3.8 Chapter Summary

In this chapter, we developed a real-time detector for identifying credential spearphishing attacks in enterprise settings. Two key contributions enabled our detector to achieve practical performance: (1) a new mental model and associated set of features that deconstructs successful spearphishing attacks into two fundamental stages, and (2) a new anomaly detection technique, DAS, that leverages these features to detect attacks, without the need for any labeled training data.

We evaluated our approach on an anonymized dataset of over 370 million emails collected from a large national laboratory that has encountered real-world attacks. At a false positive rate of less than 0.005%, our system detected all-but-two attacks in our dataset and uncovered two previously unknown successful attacks. Compared to DAS, standard anomaly detection techniques would need to generate orders of magnitude more false positives to detect the same number of attacks. Because of our approach's ability to detect a wide range of real-world spearphishing campaigns, including previously undiscovered attacks, and its low false positive cost, LBNL has implemented and deployed a version of our detector.

Part II

Mitigating Attacker Activity within the Enterprise

Chapter 4

Uncovering and Understanding Attacker Behavior within the Enterprise

4.1 Introduction

Enterprises face a range of attacks from tenacious and skilled adversaries. As such, no single defense is likely to succeed in isolation against this daunting threat landscape. Rather, the security community has long advocated for a *defense-in-depth* approach: the idea that users and organizations should layer together a set of complementary defenses to thwart attacks.

In Part I of this dissertation, we focused on developing methods that repel attackers at an enterprise's perimeter. The methods we presented for detecting spearphishing attacks provide greater protection against one of the most effective and common attacks that adversaries use to gain access to an enterprise's internal environment. Unfortunately, although the defenses proposed in our work and other related literature make successful phishing attacks significantly more difficult, skilled adversaries will nonetheless successfully compromise some of the enterprises they target.

Accordingly, Part II of this dissertation pursues a defense-in-depth strategy by examining a complementary set of defenses for thwarting sophisticated enterprise attacks. In particular, the work presented in this second part develops new insights and methods for detecting and mitigating attacker activity within an enterprise's internal environment. Our work takes aim at the following question: assuming an attacker can successfully breach an enterprise's perimeter defenses, can an organization still contain the damage caused by an attack and prevent the attack from succeeding?

We address this question by leveraging a key observation: in many real-world attacks, the initial machine or account that an adversary compromises often does not possess the functionality or data that the attacker ultimately seeks. As a result, attackers frequently need to expand their access to other machines and accounts within an enterprise to achieve their ultimate goal. This subsequent set of malicious activity, whereby an attacker spreads beyond their initial point-of-compromise to additional machines and credentials is known as *lateral movement*. Part II of this dissertation examines defenses against two prevalent strategies that attackers use to spread laterally within an enterprise: lateral phishing attacks and lateral movement between internal machines.

In the remainder of this chapter, we provide an overview of prior work on mitigating attacker lateral movement. We observe that across much of this related literature, prior work has suffered from a lack of real-world data about the traffic and activity within the environments of enterprises. Consequently, the community has lacked practical insights into the messy realities of internal enterprise activity, as well as practical defensive techniques that can operate amidst this real-world complexity. Due in part to this lack of data, the security community's focus on thwarting enterprise attacks has predominantly centered around inoculating organizations at their likely points-of-compromise, via perimeter-based defenses such as network firewalls, anti-phishing techniques, and endpoint anti-malware software.

We overcome this limitation through collaborations with two large organizations that enable us to rigorously examine internal-facing defenses against enterprise attacks. In Chapter 5, we study the problem of lateral phishing in joint work with Barracuda Networks, a large security company whose services include commercial email security protection to thousands of organizations. This chapter presents a detection strategy for uncovering attacks that attempt to spread between internal *accounts* and provides large-scale, empirical insights about the strategies that this new form of phishing employs. In Chapter 6, we develop new methods to uncover and stymie an attacker attempting to laterally move between internal *machines* within an enterprise. Evaluating our defenses on real-world data from Dropbox, we demonstrate that the new methods we introduce can detect a range of sophisticated attacks while imposing sufficiently low overhead for practical use. Taken together, the findings presented in Part II illustrate the value of hunting for and thwarting malicious activity that occurs within an organization's internal network, even after an attack's initial point-of-compromise.

Chapter 5 adapts work from a prior paper that we published at the 2019 Usenix Security Symposium [48], where it received a Distinguished Paper Award. Chapter 6 adapts work from a paper currently under submission.

4.2 Related Work and Background

In this section, we provide an overview of two categories of related work. First, we examine the literature on characterizing and mitigating lateral phishing attacks, where an attacker attempts to spread from one compromised enterprise account to additional employee accounts; this prior work relates to the findings we present in Chapter 5. Next, we survey prior work on preventing attackers from moving from one compromised enterprise machine onto additional internal machines within an organization, which relates to the work we describe in Chapter 6.

Malicious Activity in Compromised Email Accounts

Detecting Lateral Phishing Attacks: An extensive body of prior literature proposes numerous techniques for detecting traditional phishing attacks [1,5,34,37,123], as well as more sophisticated spearphishing attacks [22, 29, 60, 114, 135]. However, relatively few of these prior works focus on

lateral phishing attacks, as opposed to more traditional phishing attacks that come from external accounts.

Hu et al. studied how to use social graph metrics to detect malicious emails sent from compromised accounts [53]. Their approach detects hijacked accounts with false positive rates between 20–40%. Unfortunately, in practice, many organizations handle tens of thousands of employee-sent emails per day, so a false positive rate of 20% would lead to thousands of false alerts each day. IdentityMailer, proposed by Stringhini et al. [114], detects lateral phishing attacks by training behaviorial models based on timing patterns, metadata, and stylometry for each user. If a new email deviates from an employee's behavioral model, their system flags it as an attack. While promising, their approach produces false positive rates in the range of 1–10%, again untenable in practice given the high volume of benign emails and low base rate of phishing. Additionally, their system requires training a behavioral model for each employee, incurring expensive technical debt to operate at scale.

The system we presented in Chapter 3 can detect lateral spearphishing attacks, but the algorithms we developed rely on a set of features derived from historical user login data and enterprise network traffic logs. Organizations with less technical expertise might lack the infrastructure to comprehensively capture the enterprise's network traffic, which our prior approach requires. This technical prerequisite begs the question, can we detect lateral phishing attacks with a more minimalist dataset: only the enterprise's historical emails? Additionally, the dataset we leveraged in that chapter reflected a single enterprise that experienced only two lateral phishing attacks across a 3.5-year timespan. This small number of incidents did not enable us to characterize the general nature of lateral phishing attacks, leaving a number of gaps in our understanding of this threat, such as the prevalence of lateral phishing, common strategies that such attackers employed, and whether these attacks leverage their unique access to a legitimate account in a way that distinguishes them from external phishing emails.

Characterizing Compromised Cloud Accounts: While prior work shows that attackers frequently use phishing to compromise accounts, and that attackers occasionally conduct (lateral) phishing from these hijacked accounts, few efforts have studied the nature of lateral phishing in depth and at scale. Examining a sample of phishing emails, webpages, and compromised accounts from Google data sources, one prior study of account hijacking discovered that attackers often use these compromised accounts to send phishing emails to the account's contacts [12]. However, they concluded that automatically detecting such attacks proves challenging. Onaolapo et al. studied what attackers do with hijacked accounts [87], but they did not observe instances of lateral phishing attacks. Separate from email accounts, a study of compromised Twitter accounts found that infections appear to spread laterally through the social network. However their dataset did not allow direct observation of the lateral attack vector itself [117], nor did it provide insights into the characteristics of compromised enterprise accounts (given the nature of social media).

Taken together, prior work makes clear that cloud account compromise poses a significant and wide-spread problem, and that attackers could use their malicious access to launch lateral phishing attacks. This literature also presents promising defenses for enterprises that have sophisticated monitoring in place. Yet despite these advances, several key questions remain unresolved. Do

organizations without comprehensive monitoring and technical expertise have a practical way to defend against lateral phishing attacks? What common strategies and tradecraft do lateral phishers employ? How are lateral phishers capitalizing on their control of legitimate accounts, and what does their tactical sophistication say about the state of enterprise phishing? Chapter 5 takes a step towards answering these open questions by presenting a new detection strategy and a large-scale characterization of lateral phishing attacks.

Mitigating Lateral Movement between Machines

The prior work discussed so far focuses on how attackers can leverage a compromised email or social media account. In the remainder of this section, we review prior work that explores how organizations can mitigate the damage an attacker can cause via a compromised internal machine. Previous work pursues three general strategies for thwarting lateral movement between enterprise machines: identifying changes to an organization's security policies that increase the difficulty of lateral movement; detecting the occurrence of attacker lateral movement; and developing forensic techniques to help remediate and respond to a known attack.

Given the methods and results we present in subsequent chapters, we consider the first and last lines of prior work as complementary directions to this dissertation. In particular, our work in Chapter 6 focuses on developing practical detection for lateral movement attacks. The first direction, identifying proactive hardening and security policy changes, enables an organization to implement effective least privilege policies and identify particularly high-risk machines that warrant additional monitoring [30, 36, 42, 104]. Although these preventative measures make lateral movement more difficult, they often cannot fully eliminate all possible lateral movement paths. Indeed, our threat model (Section 6.2) assumes that organizations use these approaches to implement restrictive least privilege models, and focuses on detecting prevalent types of lateral movement attacks that can nonetheless occur. The third line of related work, forensically investigating and responding to a known attack, assumes that an organization has already identified the existence of an attack; i.e., that an enterprise has an effective detection strategy, the focus of our work in the following chapters. For example, an organization could use the forensics methods developed in prior work in conjunction with Hopper, a new system we present in Chapter 6, to effectively remediate and mitigate the damage caused by a lateral movement attack identified by our approach.

Prior work on detecting lateral movement has focused on modeling internal logins as a graph of machine-to-machine movement, and then applying traditional machine learning techniques, such as anomaly detection, to identify attacks [8, 59, 68, 69, 97, 111, 129]. Kent et al. [59] propose a supervised-learning approach for detecting compromised user credentials by training a logistic regression model to detect when an account accesses an unusual set of machines; their classifier achieves a true positive rate of 28% and incorrectly flags 1 out of every 800 users as compromised. Liu et al. [69] present Latte, a Windows-specific system for detecting lateral movement that uses a set of hard-coded Windows event codes to identify anomalodus two-hop login paths that also include a remote file execution operation after the final hop. Given a user-specified anomaly threshold, their approach can detect pentester activity on one day of their data set, but they do not report false positive numbers. log2vec [68] is a framework for transforming raw log data into

graph feature representations based on a combination of manual correlation rules and graph embedding techniques. The authors then propose a number of applications for this feature embedding framework, such as detecting lateral movement by clustering logins based on their feature embeddings and finding clusters that fall below a hand-tuned anomaly threshold; however, their system produces false positive rates between 8–10% to detect all simulated attacks in their test data sets. Similarly, Bowman et al. propose an unsupervised graph learning pipeline that implicitly clusters similar users and machines based on their prior login interactions, and then alerts on any anomalous logins. On an 18-day test dataset, their approach can detect 85% of malicious logins generated by one red team exercise at a false positive rate of 0.9% (over 100,000 false alarms on their data).

Among the best performing prior work, Siadati and Memon propose a detection system we refer to as SAL because it identifies structurally anomalous logins [111]. Evaluating their approach on one-month of real-world enterprise data, they show that SAL can detect 82% of randomly generated attack logins at a 0.3% false positive rate ($>$ 500 false alarms/day in their data set). Although prior work provides good starting points for detection, even the best of these systems generate too many false positives for practical use, and evaluate their approach on a small set of abstract, unrealistic attacks (e.g., randomly generated attack paths). As a concrete comparison, we implement SAL and describe its performance on our dataset in Section 6.7.

The work we present in Chapter 6 addresses these challenges by developing a new approach to identifying suspicious paths of logins, which we validate on a large real-world enterprise data set that includes a wide range of lateral movement attack scenarios.

Chapter 5

Detecting and Characterizing Lateral Phishing

5.1 Introduction

By and large, the high-profile coverage around targeted spearphishing attacks against major entities, such as Google, RSA, and the Democratic National Committee, has captured and shaped our mental models of enterprise phishing attacks [94, 121, 131]. In these newsworthy instances, as well as many of the targeted spearphishing incidents discussed in the academic literature [64, 65, 72], the attacks come from external accounts, created by nation-state adversaries who cleverly craft or spoof the phishing account's name and email address to resemble a known and legitimate user. However, in recent years, work from both industry [20, 62, 98] and academia [12, 49, 87, 114] has pointed to the emergence and growth of *lateral phishing* attacks: a new form of phishing that targets a diverse range of organizations and has already incurred billions of dollars in financial harm [33]. In a lateral phishing attack, an adversary uses a compromised enterprise account to send phishing emails to a new set of recipients. This attack proves particularly insidious because the attacker automatically benefits from the implicit trust in the hijacked account: trust from both human recipients and conventional email protection systems.

Although recent work presents several ideas for detecting lateral phishing [29, 38, 49, 53, 114], these prior methods either require that organizations possess sophisticated network monitoring infrastructure, or they produce too many false positives for practical usage. Moreover, prior work has not characterized this attack at scale and has not explored whether attackers use this method as a prevalent vector for lateral movement. For example, although the system we proposed in Chapter 3 can detect real-world lateral phishing attacks, our dataset from LBNL only contained two lateral phishing campaigns: an insufficiently large number to draw conclusions about the general characteristics and strategies used in these types of phishing attacks. This state of affairs leaves many important questions unanswered: How should we think about this class of phishing with respect to its scale, sophistication, and success? Do attackers follow thematic strategies, and can these common behaviors fuel new or improved defenses? How are attackers capitalizing on

the information within the hijacked accounts, and what does their behavior say about the state of enterprise phishing attacks and their potential use for lateral movement?

In this chapter, we take a first step towards answering these open questions and understanding *lateral phishing* at scale, via a research collaboration with Barracuda Networks. The work in this chapter seeks to both explore avenues for practical defenses against this burgeoning threat and develop accurate mental models for the state of these phishing attacks in the wild.

First, we present a new classifier for detecting URL-based lateral phishing emails and evaluate our approach on a dataset of 113 million emails, spanning 92 enterprise organizations. Unlike the detector proposed in Chapter 3, the classifier we construct in this chapter does not require access to the network traffic logs of an organization, which not every enterprise possesses. Instead, the new detector we propose relies solely on information contained within an email, including the raw contents of an email's message, which our detector in Chapter 3 lacked. Although the dynamic churn and dissimilarity in content across phishing emails proves challenging, the detection approach we propose can correctly identify 87.3% of attacks in our dataset, while generating less than 4 false positives per every 1,000,000 employee-sent emails.

Second, combining the attacks we detect with a corpus of user-reported lateral phishing attacks, we conduct the first large-scale characterization of lateral phishing in real-world organizations. Our analysis shows that this attack is potent and widespread: dozens of organizations, ranging from ones with fewer than 100 employees to ones with over 1,000 employees, experience lateral phishing attacks within the span of several months. In total, 14% of a set of randomly sampled organizations experienced at least one lateral phishing incident within a seven-month timespan. Furthermore, we estimate that over 11% of attackers successfully compromise at least one additional employee. Even though our ground truth sources and detector face limitations that restrict their ability to uncover stealthy or narrowly targeted attacks, our results nonetheless illuminate a prominent threat that currently affects many real-world organizations.

Examining the behavior of lateral phishers, we explore and quantify the popularity of four recipient (victim) selection strategies. Although our dataset's attackers target dozens to hundreds of recipients, these recipients often include a subset of users with some relationship to the hijacked account (e.g., fellow employees or recent contacts). Additionally, we develop a categorization for the different levels of content tailoring displayed by our dataset's phishing messages. Our categorization shows that while 7% of attacks deploy targeted messages, most attacks opt for generic content that a phisher could easily reuse across multiple organizations. In particular, we observe that lateral phishers rely predominantly on two common lures: a pretext of a shared document and a fake warning message about a problem with the recipient's account. Despite the popularity of non-targeted content, nearly one-third of our dataset's attackers invest additional time and effort to make their attacks more convincing and/or to evade detection; and, over 80% of attacks occur during the normal working hours of the hijacked account.

Ultimately, this chapter yields two contributions that expand our understanding of enterprise phishing and highlight the value of building defenses that protect against malicious activity that emanates from within an organization. First, we present a novel detector that achieves an order-of-magnitude better performance than prior work, while operating on a minimal data requirement (only leveraging historical emails). Second, through the first large-scale characterization of lateral

phishing, we uncover the scale and success of this emerging class of attacks and shed light on common strategies that lateral phishers employ. Our analysis illuminates a prevalent class of enterprise attackers whose behavior does not fully match the tactics of targeted nation-state attacks or industrial espionage. Nonetheless, these lateral phishers still achieve success in the absence of new defenses, and many of our dataset's attackers do exhibit some signs of sophistication and focused effort.

5.2 Background

In *lateral phishing* attacks, adversaries use a compromised, but legitimate, email account to send phishing emails to their victim(s). As with all phishing attacks, the attacker's goals and choice of malicious payload can take a number of different forms, from a malware-infected attachment, to a phishing URL, to a fake payment request. In this chapter, we focus on lateral phishing attacks that employ a malicious URL embedded in the email, which is the most common exploit method identified in our dataset.

Listing 5.1 shows an anonymized example of a lateral phishing attack from our study. In this attack, the phisher tried to lure the recipient into clicking on a link under the false pretense of a new contract. Additionally, the attacker also tried to make the deception more credible by responding to recipients who inquired about the email's authenticity; and they also actively hid their presence in the compromised user's mailbox by deleting all traces of their phishing email.

Lateral phishing represents a dangerous but understudied attack at the intersection of phishing and account hijacking. Phishing attacks, broadly construed, involve an attacker crafting a deceptive email from any account (compromised or spoofed) to trick their victim into performing some action. Account hijacking, also known as account takeover (ATO) in industry parlance, involves the use of a compromised account for any kind of malicious means (e.g., including spam). While prior work has primarily examined each of these attacks at a smaller scale and with respect to personal accounts, our work studies the intersection of both of these threats at a large scale and from the perspective of enterprise organizations. In doing so, we expand our understanding of lateral phishing as a potential vector for enterprise lateral movement, avenues for defending against this threat, and the strategies used by the miscreants who perpetrate these attacks.

Ethics

To conduct the work described in this chapter, we collaborated with Barracuda Networks, a large security company, to develop new detection techniques using a dataset of historical emails and reported incidents from 92 organizations who are active customers of Barracuda Networks. These organizations granted Barracuda permission to access their Office 365 employee mailboxes for the purpose of researching and developing defenses against lateral phishing. Per Barracuda's policies, all fetched emails are stored encrypted, and customers had the option of revoking access to their data at any time.

Listing 5.1: An anonymized example of a lateral phishing message that uses the lure of a fake contract document.

```
From: "Alice" <alice@company.com>
To: "Bob" <bob@company.com>
Subject: Company X (New Contract)

New Contract

View Document [this text linked to a phishing website]

Regards,
Alice [signature]
```

Due to the sensitivity of the data, only authorized employees at Barracuda during the time of our study had access to the data. No personally identifying information or sensitive data was shared with any non-employee of Barracuda. Our work also received legal approval from Barracuda, who had permission from their customers to analyze and operate on the data. Once Barracuda deployed a set of lateral phishing detectors to production, any detected attacks were reported to customers in real time to prevent financial loss and harm.

5.3 Data

In this chapter, we draw upon a dataset consisting of employee-sent emails from 92 English-language organizations; 23 organizations came from randomly sampling enterprises that had reports of lateral phishing, and 69 were randomly sampled from all organizations. Across these enterprises, 25 organizations had 100 or fewer user accounts, 34 had between 101–1000 accounts, and 33 had over 1000 accounts. Real-estate, technology, and education constituted the three most common industries in our dataset, with 15, 13, and 13 enterprises respectively; Figures 5.1 and 5.2 show the distribution of the economic sectors and sizes of our dataset's organizations, broken down by *exploratory organizations* versus *test organizations* (described below).

Schema

The organizations in our dataset use Office 365 as their email provider. At a high level, each email object contains: a unique Office 365 identifier; the email's metadata (SMTP header information), which describes properties such as the email's sent timestamp, recipients, purported sender, and subject; and the email's *body*: the contents of the email message in full HTML formatting. Office 365's documentation describes the full schema of each email object [73]. Additionally, for each organization, we have a set of *verified domains*: domains which the organization has declared that it owns.

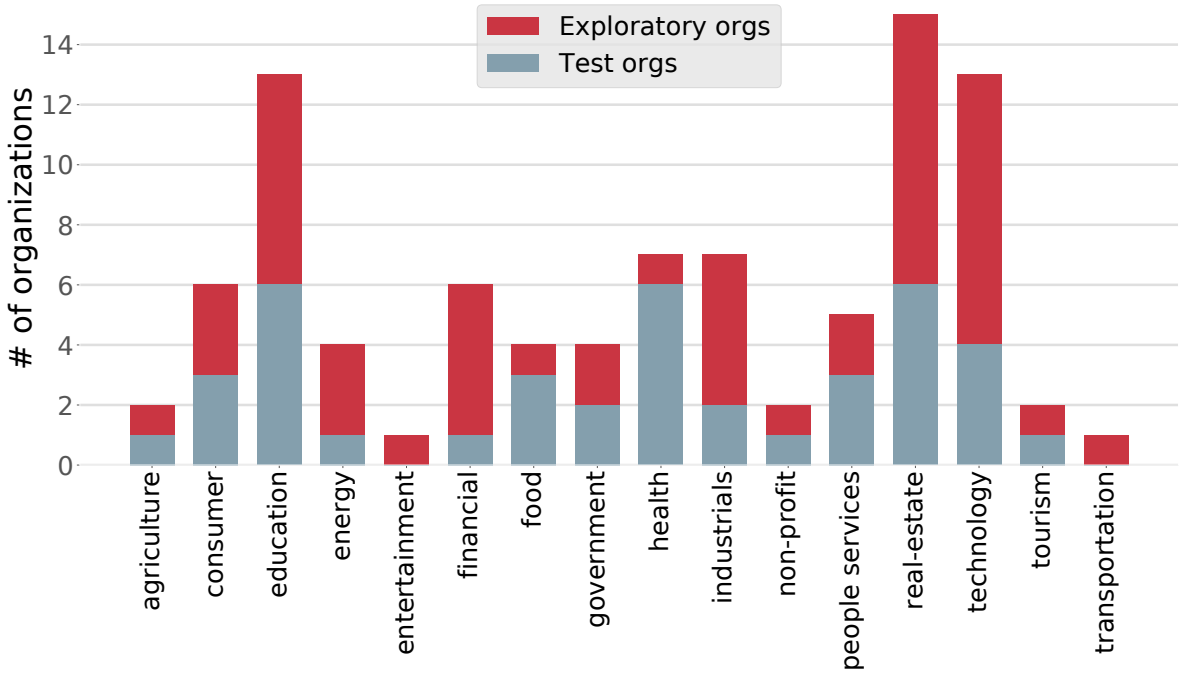

Figure 5.1: Breakdown of the economic sectors across our dataset's 52 exploratory organizations versus the 40 test organizations.

Dataset Size

Our dataset consisted of 113,083,695 unique, employee-sent emails. To ensure our detection techniques generalized (Section 5.5), we split our data into a training dataset that contained emails from 52 "exploratory organizations" during April–June 2018, and a test dataset spanning emails from July–October 2018 across 92 organizations. Our test dataset consisted of emails from the 52 exploratory organizations (but from a later, disjoint time period than our training dataset), plus data from an additional, held-out set of 40 "test organizations". We selected the 40 test organizations via a random sample that we performed prior to analyzing any data. Our training dataset had 25,670,264 emails, and our test dataset had 87,413,431 emails. Both sets of organizations covered a diverse range of industries and sizes as shown in Figures 5.1 and 5.2. The exploratory organizations spanned a total of 89,267 user mailboxes that sent or received email, and the test organizations had 138,752 mailboxes (based on the data from October 2018).[1]

Ground truth

Our set of lateral phishing emails came from two sources: (1) attack emails reported to Barracuda by an organization's security administrators, as well as attacks reported by users to their organi-

[1]The number of mailboxes is an upper bound on the number of employees due to the use of mailing lists and aliases.

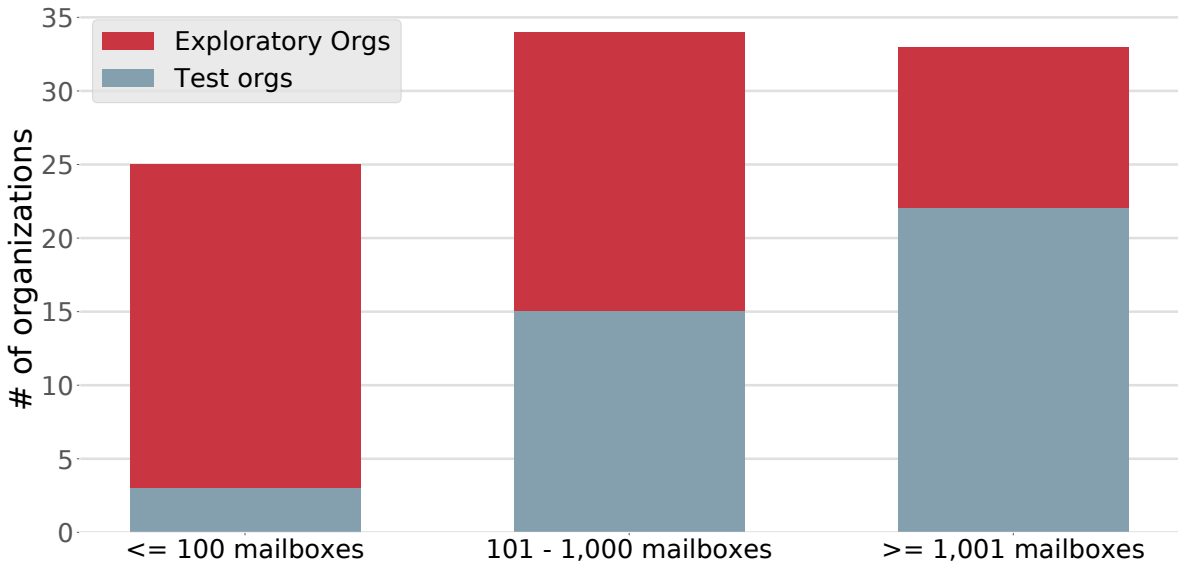

Figure 5.2: Breakdown of the organization sizes across our dataset's 52 exploratory organizations versus the 40 test organizations.

zation or directly to Barracuda, and (2) emails flagged by our detector (Section 5.4), which we manually reviewed and labeled before including.

To manually label an email as phishing, or not, we examined its message content, Office 365 metadata, and Internet Message Headers [89] to determine whether the email contained phishing content, and whether the email came from a compromised account (versus an external account, which we do not treat as lateral phishing). For example, if the Office 365 metadata showed that a copy of the email resided in the employee's Sent Items folder, or if its headers showed that the email passed the corresponding SPF or DKIM [124] checks, then we labeled the email as lateral phishing. Appendix Section B.1 describes our labeling procedure in detail.

Additionally, for a small sample of URLs in these lateral phishing emails, employees at Barracuda accessed the phishing URL in a VM-contained browser to better understand the end goals of the attack. To minimize potential harm and side effects, these employees only visited phishing URLs which contained no unique identifiers (i.e., no random strings or user/organization information in the URL path). To handle any phishing URLs that resided on URL-shortening domains, we used one of Barracuda's URL-expansion APIs that their production services already apply to email URLs, and only visited suspected phishing links that expanded to a non-side-effect URL. Most phishing URLs we explored led to a SafeBrowsing interstitial webpage, likely reflecting our use of historical emails, rather than what users would have encountered contemporaneously. However, more recent malicious URLs consistently led to credential phishing websites designed to look like a legitimate Office 365 login page (the email service provider used by our study's organizations); Figure 5.3 shows an anonymized example of one phishing website.

In total, our dataset contains 1,902 lateral phishing emails (unique by subject, sender, and sent-time), sent by 154 hijacked employee accounts from 33 organizations. 1,694 of these emails were

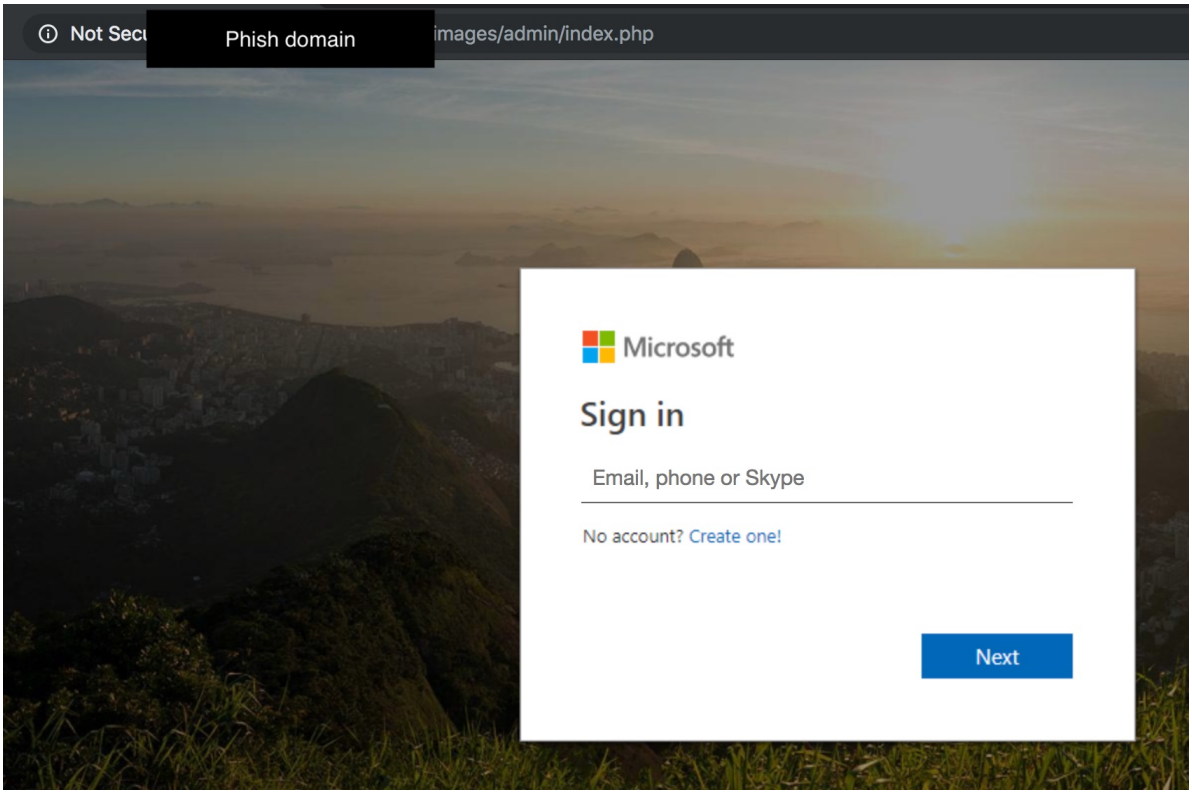

Figure 5.3: An anonymized screenshot of the web page that a lateral phishing email's URL led to.

reported by users, with the remainder found solely by our detector (Section 5.4); our detector also finds many of the user-reported attacks as well (Section 5.5). Among the user-reported attacks, 40 emails (from 12 hijacked accounts) contained a fake wire transfer or malicious attachment, while the remaining 1,862 emails used a malicious URL.

In this chapter, we focus our detection strategy on URL-based phishing, given the prevalence of this attack vector. This focus means that our analysis and detection techniques do not reflect the full space of lateral phishing attacks. Despite this limitation, our dataset's attacks span dozens of organizations, enabling us to study a prevalent class of enterprise phishing that poses an important threat in its own right.

5.4 Detecting Lateral Phishing

Adopting the *lateral attacker* threat model defined earlier in Chapter 3, we focus on phishing emails sent by a compromised employee account, where the attack embeds a malicious URL as the exploit (e.g., leading the user to a phishing webpage).

We explored three strategies for detecting lateral phishing attacks, but ultimately found that one of the strategies detected nearly all of the attacks identified by all three approaches. At a high level, the two less fruitful strategies detected attacks by looking for emails that contained (1) a rare URL and (2) a message whose text seemed likely to be used for phishing (e.g., similar text to a known phishing attack). Because our primary detection strategy detected all-but-two of the attacks found by the other strategies, while finding over ten times as many attacks, we defer discussion of the two less successful approaches to Appendix B.2; below, we focus on exploring the more effective strategy in detail. In our evaluation (Section 5.5), we report the two additional attacks found by the alternative approaches as false negatives for our detector.

Overview: We examined the user-reported lateral phishing incidents in our training dataset (April–June 2018) to identify widespread themes and behaviors that we could leverage in our detector. Grouping this set of attacks by the hijacked account (*ATO*) that sent them, we found that 95% of these ATOs sent phishing emails to 25 or more distinct recipients.[2] This prevalent behavior, along with additional feature ideas inspired by the lure-exploit detection framework introduced in Chapter 3, provide the basis for our detection strategy. In the remainder of this section, we describe the features our detector uses, the intuition behind these features, and our detector's machine learning procedure for classifying emails.

Our techniques provide neither an all-encompassing approach to finding every attack, nor guaranteed robustness against motivated adversaries trying to evade detection. However, we show in Section 5.5 that our approach finds hundreds of lateral phishing emails across dozens of real-world organizations, while incurring a low volume of false positives.

Features: Our detector extracts three sets of features. The first set consists of two features that target the popular behavior we observed earlier: contacting many recipients. Given an email, we first extract the number of unique recipients across the email's `To`, `CC`, and `BCC` headers. Additionally, we compute the Jaccard similarity of this email's recipient set to the closest set of historical recipients seen in any employee-sent email from the preceding month. We refer to this latter (similarity) feature as the email's recipient likelihood score.

The next two sets of features draw upon the lure-exploit phishing framework described in Section 3.2. Recall that this framework posits that phishing emails contain two necessary components: a "lure", which convinces the victim to believe the phishing email and perform some action; and an "exploit": the malicious action the victim should execute.

To characterize whether a new email contains a potential phishing lure, our detector extracts a single, lightweight boolean feature based on the email's text. Specifically, Barracuda provided us with a set of roughly 150 keywords and phrases that frequently occur in phishing attacks. They

[2]To assess the generalizability of our approach, our evaluation uses a withheld dataset, from a later timeframe and with new organizations (Section 5.5).

developed this set of "phishy" keywords by extracting the link text from several hundred real-world phishing emails (both external and lateral phishing) and selecting the (normalized) text that occurred most frequently among these attacks. Thematically, these suspicious keywords convey a call to action that entices the recipient to click a link. For our "lure" feature, we extract a boolean value that indicates whether an email contains any of these phishy keywords.

Finally, we complete our detector's feature set by extracting two features that capture whether an email might contain an exploit. Since our work focuses on URL-based attacks, this set of features reflects whether the email contains a potentially dangerous URL.

First, for each email, we extract a *global URL reputation* feature that quantifies the rarest URL an email contains. Given an email, we extract all URLs from the email's body and ignore URLs if they fall under two categories: we exclude all URLs whose domain is listed on the organization's *verified domain* list (Section 5.3), and we also exclude all URLs whose displayed, hyperlinked text exactly matches the URL of the hyperlink's underlying destination. For example, in Listing 5.1's attack, the displayed text of the phishing hyperlink was "Click Here", which does not match the hyperlink's destination (the phishing site), so our procedure would keep this URL. In contrast, Alice's signature from Listing 5.1 might contain a link to her personal website, e.g., `www.alice.com`; our procedure would ignore this URL, since the displayed text of `www.alice.com` matches the hyperlink's destination.

This latter filtering criteria makes the assumption that a phishing URL will attempt to obfuscate itself, and will not display the true underlying destination directly to the user. After these filtering steps, we extract a numerical feature by mapping each remaining URL to its registered domain, and then looking up each domain's ranking on the Cisco Umbrella Top 1 Million sites [54];[3] for any unlisted domain, we assign it a default ranking of 10 million. We treat two special cases differently. For URLs on shortener domains, our detector attempts to recursively resolve the shortlink to its final destination. If this resolution succeeds, we use the global ranking of the final URL's domain; otherwise, we treat the URL as coming from an unranked domain (10 million). For URLs on content hosting sites (e.g., Google Drive or Sharepoint), we have no good way to determine its suspiciousness without fetching the content and analyzing it (an action that has several practical hurdles). As a result, we treat all URLs on content hosting sites as if they reside on unranked domains.

After ranking each URL's domain, we set the email's *global URL reputation* feature to be the worst (highest) domain ranking among its URLs. Intuitively, we expect that phishers will rarely host phishing pages on popular sites, so a higher *global URL reputation* indicates a more suspicious email. In principle a motivated adversary could evade this feature; e.g., if an adversary can compromise one of the organization's verified domains, they can host their phishing URL from this compromised site and avoid an accurate ranking. However, we found no such instances in our set of user-reported lateral phishing. Additionally, since the goal of this work is to begin exploring practical detection techniques, and develop a large set of lateral phishing incidents for our analysis, this feature suffices for our needs.

[3] We use a list fetched in early March 2018 for our feature extraction, but in practice, one could use a continuously updated list.

In addition to this global reputation metric, we extract a local metric that characterizes the rareness of a URL with respect to the domains of URLs that an organization's employees typically send. Given a set of URLs embedded within an email, we map each URL to its fully-qualified domain name (FQDN) and count the number of days from the preceding month where at least one employee-sent email included a URL on the FQDN. We then take the minimum value across all of an email's URLs; we call this minimum value the *local URL reputation* feature. Intuitively, suspicious URLs will have both a low global reputation and a low local reputation. However, our evaluation (Section 5.5) finds that this *local URL reputation* feature adds little value: URLs with a low *local URL reputation* value almost always have a low *global URL reputation* value, and vice versa.

Classification: To label an email as phishing or not, we trained a Random Forest classifier [127] with the aforementioned features. To train our classifier, we take all user-reported lateral phishing emails in our training dataset, and combine them with a set of likely-benign emails. We generate this set of "benign" emails by randomly sampling a subset of the training window's emails that have not been reported as phishing; we sample 200 of these benign emails for each attack email to form our set of benign emails for training. Following standard machine learning practices, we selected both the hyperparameters for our classifier and the exact downsampling ratio (200:1) using cross-validation on this training data. Appendix B.1 describes our training procedure in more detail.

Once we have a trained classifier, given a new email, our detector extracts its features, feeds the features into this classifier, and outputs the classifier's decision.

5.5 Evaluation

In this section we evaluate our lateral phishing detector. We first describe our testing methodology, and then show how well the detector performs on millions of emails from over 90 organizations. Overall, our detector has a high detection rate, generates few false positives, and detects many new attacks.

Methodology

Establishing Generalizability: As described earlier in Section 5.3, we split our dataset into two disjoint segments: a *training dataset* consisting of emails from the 52 exploratory organizations during April–June 2018 and a *test dataset* from 92 enterprises during July–October 2018; as shown later in this section, our detector's performance remains the same if our test dataset contains only the emails from the 40 withheld test organizations. Given these two datasets, we first trained our classifier and tuned its hyperparameters via cross validation on our training dataset (Appendix B.1). Next, to compute our evaluation results, we ran our detector on each month of the held-out test dataset. To simulate a classifier in production, we followed standard machine learning practices and used a continuous learning procedure to update our detector each month [102]. Namely, at the

end of each month, we aggregated the user-reported and detector-discovered phishing emails from all previous months into a new set of phishing "training" data; and, we aggregated our original set of randomly sampled benign emails with our detector's false positives from all previous months to form a new benign "training" dataset. We then trained a new model on this aggregated training dataset and used this updated model to classify the subsequent month's data. However, to ensure that any tuning or knowledge we derived from the training dataset did not bias or overfit our classifier, we did not alter any of the model's hyperparameters or features during our evaluation on the test dataset.

Our evaluation's temporal-split between the training and test datasets, along with the introduction of new data from randomly withheld organizations into the test dataset, follows best practices that recommend this approach over a randomized cross-validation evaluation [3, 76, 90]. A completely randomized evaluation (e.g., cross-validation) risks training on data from the future and testing on the past, which might lead us to overestimate the detector's effectiveness. In contrast, our methodology evaluates our detector with fresh data from a "future" time period and introduces 40 new organizations, neither of which our detector saw during training time; this also reflects how a detector operates in practice.

Alert Metric (Incidents): We have several choices for modeling our detector's alert generation process (i.e., how we count *distinct* attacks). For example, we could evaluate our detector's performance in terms of how many unique emails it correctly labels. Or, we could measure our detector's performance in terms of how many distinct employee accounts it marks as compromised (modeling a detector that generates one alert per account and suppresses the rest). Ultimately, we select a notion commonly used in practice, that of an *incident*, which corresponds to a unique (subject, sender email address) pair. At this granularity, our detector's alert generation model produces a single alert per unique (subject, sender) pair. This metric avoids biased evaluation numbers that overemphasize compromise incidents that generate many identical emails during a single attack. For example, if there are two incidents, one which generates one hundred emails to one recipient each, and another which generates one email to 100 recipients, a detector's performance on the hundred-email incident will dominate the result if we count attacks at the email level.

In total, our training dataset contains 40 lateral phishing incidents from our user-reported ground truth sources, and our test dataset contains 61 user-reported incidents. Our detector finds an additional 77 unreported incidents, as reported in Row 2 of Table 5.1.

Detection Results

Table 5.1 summarizes the performance metrics for our detector. We use the term *Detection Rate* to refer to the percentage of lateral phishing incidents that our detector finds, divided by all known attack incidents in our dataset (i.e., any user-reported incident and any incident found by any detection technique we tried). For completeness, we include the 12 attachment-based incidents in our False Negative and Detection Rate computations, which our detector obviously misses since we designed it to catch URL-based lateral phishing. Additionally, we also include, as false negatives, 2 training incidents that our less successful detectors identified (Appendix B.2); these two alternative

Metric	Training April – June 2018	Testing July – October 2018
Organizations	52 Exploratory	52 Exploratory + 40 Test
Detected Known Attacks	34	47
Detected New Attacks	28	49
Missed Attacks (FN)	8	14
Detection Rate	88.6%	87.3%
Total Emails	25,670,264	87,413,431
False Positives (FP)	136	316
False Positive Rate	0.00053%	0.00036%
Precision	31.3%	23.3%

Table 5.1: Evaluation results of our detector. "Detected Known Attacks" shows the number of incidents that our detector identified, and were also reported by an employee at an organization. "Detected New Attacks" shows the number of incidents that our detector identified, but were not reported by anyone. "Missed Attacks (FN)" shows all incidents either reported by a user or found by any of our detection strategies, but our detector marked it as benign (false negative). Of the 22 incidents our detector misses, 12 are attachment-based attacks, a threat model which our detector explicitly does not target but which we include in our FN and Detection Rate results for completeness.

strategies did not find any new attacks in the test dataset. Thus, the *Detection Rate* reflects a best-effort assessment that potentially overestimates the true positive rate of our detector, since we have an imperfect ground truth that cannot account for narrowly targeted attacks that go unreported by users. *Precision* equals the percent of attack alerts (incidents) produced by our detector divided by the total number of alerts our detector generated (attacks plus false positives).

Training and Tuning: On the training dataset, our detector correctly identified 62 out of 70 lateral phishing incidents (88.6%), while generating a total of 62 false positives across 25.7 million employee-sent emails.

Our PySpark Random Forest classifier exposes a built-in estimate of each feature's relative importance [112], where each feature receives a score between 0.0–1.0 and the sum of all the scores adds up to 1.0. Based on these feature weights, our model places the most emphasis on the *global URL reputation* feature, giving it a weight of 0.42, and the email's "number of recipients" feature (0.34). In contrast, our model essentially ignores our *local URL reputation*, assigning it a score of 0.01, likely because most globally rare domains tend to also be locally rare. Of the remaining features, the recipient likelihood feature has a weight of 0.17 and the "phishy" keyword feature has a weight of 0.06.

Test Dataset: Our detector correctly identified 96 lateral phishing incidents out of the 110 test incidents (87.3%) across our ground truth dataset. Additionally, our detector discovered 49 incidents that, according to our ground truth, were not reported by a user as phishing. With respect to its cost, our detector generated 312 total false positives across the entire test dataset (a false positive

rate of less than 0.00035%, assuming that emails not identified as an attack by our ground truth are benign). Across our test dataset, 82 out of the 92 organizations accumulated 10 or fewer false positives across the entire four month window, with 44 organizations encountering zero false positives across this timespan. In contrast, only three organizations had more than 40 total false positives across all four months (encountering 44, 66, and 83 false positives, respectively). Our detector achieves similar results if we evaluate on just the data from our 40 withheld test organizations, with a Detection Rate of 91.0%, a precision of 23.1%, and a false positive rate of 0.00038%.

Bias and Evasion: We base our evaluation numbers on the best ground truth we have: a combination of all user-reported lateral phishing incidents (including some attacks outside our threat model), and all incidents discovered by any detection technique we tried (which includes two approaches orthogonal to our detector's strategy). This ground truth suffers from a bias towards phishing emails that contact many potential victims, and attacks that users can more easily recognize. Additionally, since our detector focuses on URL-based exploits, our dataset of attacks likely underestimates the prevalence of non-URL-based phishing attacks, which come solely from user-reported instances in our dataset. As a result, our work does not capture the full space of lateral phishing attacks, such as ones where the attacker targets a narrow, select set of victims with stealthily deceptive content. Rather, given that our detector identifies many known and unreported attacks, while generating only a few false positives per month, we provide a starting point for practical detection that future work can extend. Moreover, even if our detector does not capture every possible attack, the fact that the attacks in our dataset span dozens of different organizations, across a multi-month timeframe, allows us to illuminate a class of understudied attacks that many enterprises currently face.

Aside from obtaining more comprehensive ground truth, more work is needed to explore defenses against potential evasion attacks. Attackers could attempt to evade our detector by targeting different features we draw upon, such as the composition or number of recipients they target. Against many of these evasion attacks, future work could leverage additional features and data, such as the actions a user takes within an email account (e.g., reconnaissance actions, such as unusual searches, that indicate an attacker mining the account for targeted recipients to attack) or information from the user's account log-on (e.g., the detector we developed in Chapter 3 leverages an account's login IP address to detect lateral phishing). At the same time, future work should study which evasion attacks remain economically feasible for attackers to conduct. For example, an attacker could choose to only target a small number of users in the hopes of evading our detector; but even if this evasion succeeded, the conversion rate of fooling a recipient might be so low that the attack ultimately fails to compromise an economically viable number of victims. Indeed, as we explore in the following section (Section 5.6), the attackers captured in our dataset already engage in a range of different behaviors, including a few forms of sophisticated, manual effort to increase the success of their attacks.

Scale and Success	
# distinct phishing emails	1,902
# incidents	180
# ATOs	154
# organizations w/ 1+ incident	33
# phishing recipients	101,276
% successful ATOs	11%
# employee recip (average) for compromise	542

Table 5.2: Summary of the scale and success of the lateral phishing attacks in our dataset (Section 5.6).

5.6 Characterizing Lateral Phishing

In this section, we present an analysis of real-world lateral phishing using all known attacks across our entire dataset (both training and test). During the seven month timespan, a total of 33 organizations experienced lateral phishing attacks, with the majority of these compromised organizations experiencing multiple incidents. Examining the thematic message content and recipient targeting strategies of the attacks, our analysis suggests that most lateral phishers in our dataset do not actively mine a hijacked account's emails to craft personalized spearphishing attacks. Rather, these attackers operate in an opportunistic fashion and rely on commonplace phishing content. This finding suggests that the space of enterprise phishing has expanded beyond its historical association with sophisticated APTs and nation-state adversaries.

At the same time, these attacks nonetheless succeed, and a significant fraction of attackers do exhibit some signs of sophistication and attention to detail. As an estimate of the success of lateral phishing attacks, at least 11% of our dataset's attackers successfully compromise at least one other employee account. In terms of more refined tactics, 31% of lateral phishers invest some manual effort in evading detection or increasing their attack's success rate. Additionally, over 80% of the attacks in our dataset occur during the normal working hours of the hijacked account. Taken together, our results suggest that lateral phishing attacks pose a prevalent enterprise threat that real-world attackers use to expand their malicious access.

In addition to exploring attacks at the incident granularity (as done in Section 5.5), this section also explores attacks at the granularity of a lateral phisher (hijacked account) when studying different attacker behaviors. As described in Section 5.2, industry practitioners often refer to such hijacked accounts as ATOs, and throughout this section, we use the terms *hijacked account*, *lateral phisher*, and *ATO* synonymously.

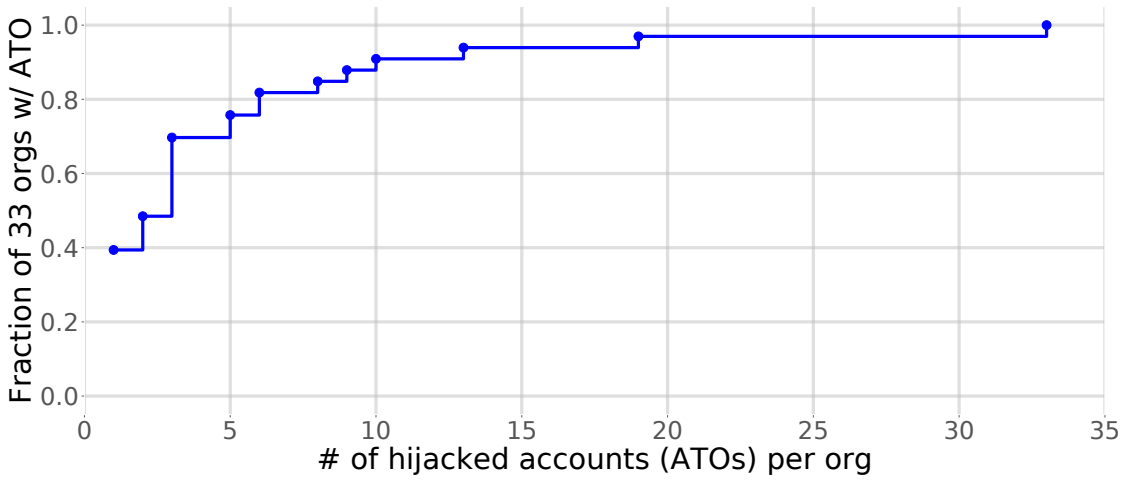

Figure 5.4: Fraction of organizations with x hijacked accounts that sent at least one lateral phishing email. 13 organizations had only 1 ATO; the remaining 20 saw lateral phishing from 2+ ATOs (Section 5.6).

Scale and Success of Lateral Phishing

Scale: Our dataset contains 1,902 distinct lateral phishing emails sent by 154 hijacked accounts.[4] A total of 33 organizations in our dataset experience at least one lateral phishing incident: 23 of these organizations came from sampling the set of enterprises with known lateral phishing incidents (Section 5.3), while the remaining 10 came from the 69 organizations we sampled from the general population. Assuming our random sample reflects the broader population of enterprises, over 14% of organizations experience at least one lateral phishing incident within a 7 month timespan. Furthermore, based on Figure 5.4, over 60% of the compromised organizations in our dataset experienced lateral phishing attacks from at least two hijacked employee accounts. Given that our set of attacks likely contains false negatives (thus underestimating the prevalence of attacks), these numbers illustrate that lateral phishing attacks are widespread across enterprise organizations.

Successful Attacks: Given our dataset, we do not definitively know whether an attack succeeded. However, we conservatively (under)estimated the success rate of lateral phishing using the methodology below. Based on this procedure, we estimate that at least 11% of lateral phishers successfully compromise at least one new enterprise account.

Let Alice and Bob represent two different ATOs at the same organization, where P_A and P_B represent one of Alice's and Bob's phishing emails respectively, and $Reply_B$ represents a reply from Bob to a lateral phishing email he received from Alice. Intuitively, our methodology concludes that Alice successfully compromised Bob if (1) Bob received a phishing email from Alice, (2) shortly after receiving Alice's phishing email, Bob then subsequently sent his own phishing email, and (3) we have strong evidence that the two employees' phishing emails are related (reflected in criteria 3 and 4 below).

[4] A distinct phishing email contains a fully unique tuple of (sender, subject, timestamp, and recipients) that does not match any other email.

Formally, we say that P_A succeeded in compromising Bob's account if *all* of the following conditions are true:

1. Bob was a recipient of P_A
2. After receiving P_A, Bob subsequently sent his own lateral phishing emails (P_B)
3. Either of the following two conditions are met:
    a) P_B and P_A used similar phishing content: if the two attacks used identical subjects or if both of the phishing URLs they used belonged to the same fully-qualified domain
    b) Bob sent a reply ($Reply_B$) to P_A, where his reply suggests he fell for Alice's attack and where Bob sent $Reply_B$ prior to his own attack (P_B)
4. Either of the following two conditions are met:
    a) P_B was sent within two days after Bob received P_A
    b) P_B and P_A used identical phishing messages or their phishing URLs' paths followed nearly identical structures (e.g., '`http://X.com/z/office365/index.html`' vs. '`http://Y.com/z/office365/index.html`')

Unpacking the final criteria (#4), in the first case (4.a), we settled on a two-day interarrival threshold based on prior literature [57, 58], which suggests that 50% of users respond to an email within 2 days and roughly 75% of users who click on a spam email do so within 2 days. Assuming that phishing follows similar time constants for how long it takes a recipient to take action, 2 days represented a conservative threshold to establish a link between P_A and P_B. At the same time, both prior works show there exists a long tail of users who take weeks to read and act on an email. The second part (4.b) attempts to address this long tail by raising the similarity requirements between Alice and Bob's attacks before concluding that former caused the latter. For successful attackers labeled by heuristic 4.b, the longest observed time gap between P_A and P_B is 17 days, which falls within a plausible timescale based on the aforementioned literature.

From this methodology, we conclude that 17 ATOs successfully compromised at least 23 new enterprise accounts. While our procedure might erroneously identify cases where an attacker has concurrently compromised both Alice and Bob (rather than compromising Bob's account via Alice's), the first two criteria, requiring Bob to be a *recent recipient* of Alice's phishing email, help reduce this error. Our procedure likely underestimates the general success rate of lateral phishing attacks, since it does not identify successful attacks where the attacker does not subsequently use Bob's account to send phishing emails, nor does it account for false negatives in our dataset or attacks outside of our visibility (e.g., compromise of recipients at external organizations).

Recipient Targeting and Conversion Rate

In this section, we estimate the conversion rate of our dataset's lateral phishing attacks, and discuss four recipient targeting strategies that reflect the behavior of most attackers in our dataset. Our

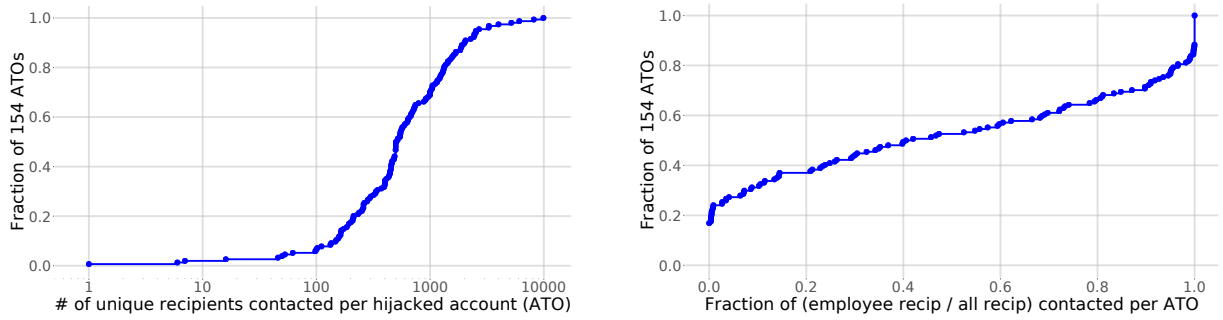

Figure 5.5: The left CDF shows the distribution of the total number of phishing recipients per ATO. The right CDF shows the fraction of ATOs where $x\%$ of their total recipient set consists of fellow employees.

analysis suggests that attackers sought to spread laterally throughout their victim's organizations in over half of the hijacked accounts in our dataset. However, given the large numbers of recipients targeted by most of these attacks, the use of lateral phishing in our dataset appears to reflect the tradecraft of opportunistic attackers, who might not have the time or skill to engage in more sophisticated lateral movement methods (such as the ones we describe later in Chapter 6).

Recipient Volume and Estimated Conversation Rate: Cumulatively, the lateral phishers in our dataset contact 101,276 unique recipients, where 41,740 belong to the same organization as the ATO. As shown in Figure 5.5, more than 94% of the attackers send their phishing emails to over 100 recipients; with respect to the general population of all lateral phishers, this percentage likely overestimates the prevalence of high "recipient-volume" attackers, since our detector draws on recipient-related features.

Targeting hundreds of people gives attackers a larger pool of potential victims, but it also incurs a risk that a recipient will detect and flag the attack either to their security team or their fellow recipients (e.g., via Reply-All). To isolate their victims and minimize the ability for fellow recipients to warn each other, we found that attackers frequently contact their recipients via a mass `BCC` or through many individual emails.

Aside from this containment strategy, we also estimate that our dataset's lateral phishing attacks have a difficult time fooling an individual employee, and thus might require targeting many recipients to hijack a new account. Earlier in this section, we found that 17 ATOs successfully compromised 23 new accounts. Looking at the number of accounts they successfully hijacked divided by the number of fellow employees they targeted, the median conversation rate for our attackers was 1 newly hijacked account per 542 fellow employees; the attacker with the best conversation rate contacted an average of 26 employees per successful compromise. We caution that our method for determining whether an attack succeeded (Section 5.6) does not cover all cases, so our conversation rate might also underestimate the success of these attacks in practice. But if our estimated conversion rate accurately approximates the true rate, it would explain why these attackers contact so many recipients, despite the increased risk of detection.

Recipient Targeting Strategies: Anecdotally, we know that some lateral phishers select their set of victims by leveraging information in the hijacked account to target familiar users; for example,

Recipient Targeting Strategy	# ATOs
Account-agnostic	63
Organization-wide	39
Lateral-organization	2
Targeted-recipient	44
Inconclusive	6

Table 5.3: Summary of recipient targeting strategies per ATO (Section 5.6).

sending their attack to a subset of the account's "Contact Book". Unfortunately our dataset does not include information about any reconnaissance actions that an attacker performed to select their phishing recipients (e.g., explicitly searching through a user's contact book or recent recipients).

Instead, we empirically explore the recipient sets across our dataset's attackers to identify plausible strategies for how these attackers might have chosen their set of victims. Four recipient targeting strategies, summarized in Table 5.3 and explained below, reflect the behavior of all but six attackers in our dataset. To help assess whether a recipient and the ATO share a meaningful relationship, we compute each ATO's *recent contacts*: the set of all email addresses whom the ATO sent at least one email to in the 30 days preceding the ATO's phishing emails. While some ATOs (28.6%) specifically target many of their account's recent contacts, the majority of these lateral phishers appear more interested in either contacting many arbitrary recipients or sending phishing emails to a large fraction of the hijacked account's organization.

Account-agnostic Attackers: Starting with the least-targeted behavior, 63 ATOs in our dataset sent their attacks to a wide range of recipients, most of whom do not appear closely related to the hijacked account. We call this group *Account-agnostic attackers*, and identify them using two heuristics.

First, we categorize an attacker as Account-agnostic if less than 1% of the recipients belong to the same organization as the ATO, and further exploration of their recipients does not reveal a strong connection with the account. Examining the right-hand graph in Figure 5.5, 37 ATOs target recipient sets where less than 1% of the recipients belong to the same organization as the ATO. To rule out the possibility that these attackers' recipients are nonetheless related to the account, we computed the fraction of recipients who appeared in each ATO's recent contacts; for all of the 37 possible Account-agnostic ATOs, less than 17% of their attack's total recipients appeared in their recent contacts. Among these 37 candidate Account-agnostic ATOs, 33 of them contact recipients at 10 or more organizations (unique recipient email domains), 2 of them exclusively target either Gmail or Hotmail accounts, and the remaining 2 ATOs are best described as Lateral-organization attackers (described below).[5] Excluding the 2 Lateral-organization attackers, the 35 ATOs identified by this first criteria sent their attacks to predominantly external recipients, belonging to either many different organizations or exclusively to personal email hosting services (e.g., Gmail and

[5]Figure B.6 in Appendix B.5 shows the distribution of recipient domains contacted by all ATOs.

Hotmail), and only a small percentage of these recipients appeared in the ATO's recent contacts; as such, we label these 35 attackers as Account-agnostic.

Second, we expand our search for Account-agnostic attackers by searching for attackers where less than 50% of the ATO's total recipients also belong to the ATO's organization, and where the ATO contacts recipients at many different organizations; specifically, where the ATO's phishing recipients belonged to over twice as many unique domains as all of the email addresses in ATO's recent contacts. This search identified 63 ATOs. To filter out attackers in this set who may have drawn on the hijacked account's recent contacts, we exclude any ATO where over 17% of their attack's total recipients also appeared in the ATO's recent contacts (17% was the maximum percentage among ATOs from the first Account-agnostic heuristic). After applying this last condition, our second heuristic identifies 54 Account-agnostic attackers.

Combining and deduplicating the ATOs from both criteria results in a total of 63 Account-agnostic attackers (40.9%): lateral phishers who predominantly target recipients without close relationships to the hijacked account or its organization.

Lateral-organization Attackers: During our exploration of potential Account-agnostic ATOs, we uncovered 2 attackers whom we label under a different category: *Lateral-organization attackers*. In both these cases, less than 1% of the attacker's recipients belonged to the same organization as the ATO, but each attacker's recipients did belong to organizations within the same industry as the ATO's organization. This thematic characteristic among the recipients suggests a deliberate strategy to spread across organizations within the targeted industries, so accordingly, we categorize them as Lateral-organization attackers.

Organization-wide Attackers: Office 365 provides a "Groups" feature that lists the different groups that an account belongs to [74]. For some enterprises, this feature enumerates most, if not all, employees at the organization. Thus, lateral phishers who wish to cast a wide phishing net might adopt a simple strategy of sending their attack to everyone at the organization. We call these ATOs *Organization-wide attackers* and identify them through two ways.

First, we search for any attackers where at least half of their phishing recipients belong to the ATO's organization, and where at least 50% of the organization's employees received the phishing email (i.e., the majority of a phisher's victims were employees and the attacker targeted a majority of the enterprise); this search yielded a total of 16 ATOs. We estimate the list of an organization's employees by building a set of all employee email addresses who sent or received email from anyone during the entire month of the phishing incident.[6] For all of these 16 ATOs, less than 11% of the recipients they target also appear in their recent contacts. Coupled with the fact that each of these ATOs contacts over 1,300 recipients, their behavior suggests that their initial goal focuses on phishing as many of the enterprise's recipients as possible, rather than targeting users particularly close to the hijacked account. Accordingly, we categorize them as Organization-wide attackers.

Our second heuristic looks for attackers whose recipient set consists nearly entirely of fellow employees, but where the majority of the organization does not necessarily receive a phishing email. Revisiting Figure 5.5, 36 candidate Organization-wide ATOs sent over 95% of their phish-

[6]This collection likely overestimates the actual set of employees because of service addresses, mailing list aliases, and personnel churn.

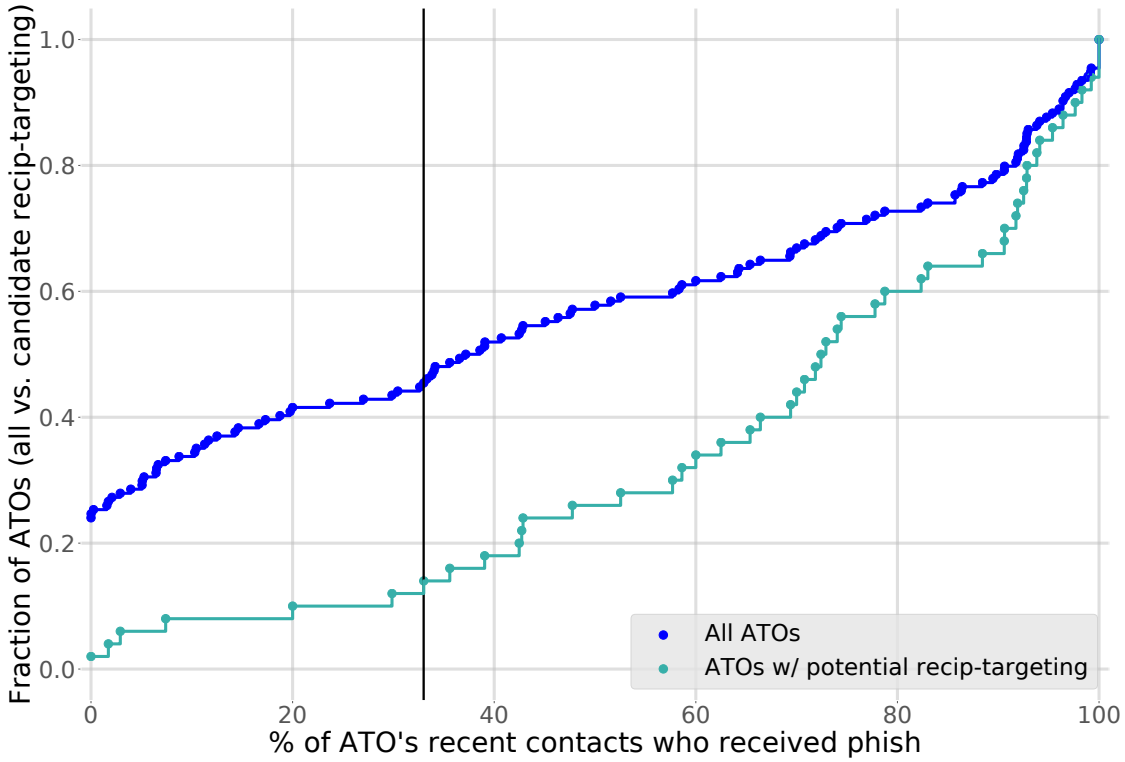

Figure 5.6: CDF: the x-axis displays what % of the ATO's recent contacts received a lateral phishing email (Section 5.6). The bottom teal graph filters the ATOs to *exclude* any ATO identified as Account-agnostic, Lateral-organization, and Organization-wide attackers; at the vertical black line, 88% of these filtered ATOs send phishing emails to at least $x = 33\%$ of addresses from their recent contacts.

ing emails to fellow employee recipients. However, we again need to exclude and account for ATOs who leverage their hijacked account's recent contacts. From the first Organization-wide heuristic discussed previously, we saw that less than 11% of the recipients of that heuristic's Organization-wide attackers came from the ATO's recent contacts. Using this value as a final threshold for this second candidate set of Organization-wide attackers, we identify 29 Organization-wide attackers where over 95% of their recipients belong to the ATO's organization but less than 11% of the recipients came from the ATO's recent contacts; a combination that suggests the attacker seeks primarily to compromise other employees, but who do not necessarily have a personal connection with the hijacked account.

Aggregating and deduplicating the two sets of lateral phishers from above produces a total of 39 Organization-wide attackers (25.3%), who target many fellow employees with their phishing attacks.

Targeted-recipient Attackers: For the remaining, uncategorized 50 ATOs, we cannot conclusively determine the attackers' recipient targeting strategies because our dataset does not provide us with the full set of information and actions available to the attacker. Nonetheless, Figure 5.6

presents some evidence that 44 of these remaining attackers do draw upon the hijacked account's prior relationships. Specifically, 44 attackers sent their attacks to at least 33% of the addresses in the ATO's recent contacts.[7] Since these ATOs sent attacks to at least 1 out of every 3 of the ATO's recently contacted recipients, these attackers appear interested in targeting a substantial fraction of users with known ties to the hijacked account. As such, we label these 44 ATOs as Targeted-recipient attackers.

Message Content: Tailoring and Themes

Since lateral phishers control a legitimate employee account, these attackers could easily mine recent emails to craft personalized spearphishing messages. To understand how much attackers do leverage their privileged access in their phishing attacks, this section characterizes the level of tailoring we see among lateral phishing messages. Overall, only 7% of our dataset's incidents contain targeted content within their messages. Across the phishing emails that used non-targeted content, the attackers in our dataset relied on two predominant narratives (deceptive pretexts) to lure their victim into performing a malicious action. The combination of these two results suggests that, for the present moment, these attackers (across dozens of organizations) see more value in opportunistically phishing as many recipients as possible, rather than investing time to mine the hijacked accounts for personalized spearphishing fodder.

Content Tailoring: When analyzing the phishing messages in our dataset, we found that two dimensions aptly characterized the different levels of content tailoring and customization. The first dimension, "Topic tailoring", describes how personalized the topic or main idea of the email is to the victim or organization. The second dimension, "Name tailoring", describes how specifically the attacker addresses the victim (e.g., "Dear user" vs. "Dear Bob"). For each of these two dimensions, we enumerate three different levels of tailoring and provide an anonymized message snippet below; we use Bob to refer to one of the attack's recipients and FooCorp for the company that Bob works at.

1. Topic tailoring: the uniqueness and relevancy of the message's topic to the victim or organization:

    a) Generic phishing topic: an unspecific message that could be sent to any user ("You have a new shared document available.")

    b) Broadly enterprise related topic: a message that appears targeted to enterprise environments, but one that would also make sense if the attacker used it at many other organizations ("Updated work schedule. Please distribute to your teams.")

[7] When examining and applying thresholds for the Account-agnostic and Organization-wide Attackers, we used a slightly different fraction: how many of the ATO's phishing recipients also appeared in their recent contacts? Here, we seek to capture attackers who make a specific effort to target a considerable number of familiar recipients. Accordingly, we look at the fraction of the ATO's recent contacts that received phishing emails, where the denominator reflects the number of users in the ATO's recent contacts, rather than the ATO's total number of phishing recipients.

	Generic	Enterprise	Targeted
No naming	90	35	9
Organization named	23	16	4
Recipient named	0	3	0

Table 5.4: Distribution of the number of *incidents* per message tailoring category (Section 5.6). The columns correspond to how unique and specific the message's topic pertains to the victim or organization. The rows correspond to whether the phishing email explicitly names the recipient or organization.

c) Targeted topic: a message where the topic clearly relies on specific details about the recipient or organization ("Please see the attached announcement about FooCorp's 25th year anniversary.", where FooCorp has existed for exactly 25 years.)

2. Name tailoring: whether the phishing message specifically uses the recipient or organization's name:

a) Non-personalized naming: the attack does not mention the organization or recipient by name ("Dear user, we have detected an error in your mailbox settings...")

b) Organization specifically named: the attack mentions just the organization, but not the recipient ("New secure email message from FooCorp...")

c) Recipient specifically named: the attack specifically uses the victim's name in the email ("Bob, please review the attached purchase order...")

Taken together, this taxonomy divides phishing content into nine different classes of tailoring; Table 5.4 shows how many of our dataset's 180 incidents fall into each category. From this categorization, two interesting observations emerge. First, only 3 incidents (1.7%) actually addressed their recipients by name. Since most ATOs (94%) in our dataset emailed at least 100 recipients, attackers would need to leverage some form of automation to both send hundreds of individual emails and customize the naming in each one. Based on our results, it appears these attackers did not view that as a worthwhile investment. For example, they might fear that sending many individual emails might trigger an anti-spam or anti-phishing mechanism, which we observed in the case of one ATO who attempted to send hundreds of individual emails. Second, looking at the last column of Table 5.4, only 13 incidents (7%) used targeted content in their messages. The overwhelming majority (92.7%) of incidents opted for more generic messages that an attacker could deploy at a large number of organizations with minimal changes (e.g., by only changing the name of the victim organization).

While our attack dataset captures a limited view of all lateral phishing attacks, it nonetheless reflects all known lateral phishing incidents across 33 organizations over a 7-month timeframe. Thus, despite the data's limitations, our results show that a substantial fraction of lateral phishers do not fully draw upon their compromised account's resources (i.e., historical emails) to craft personalized spearphishing messages. This finding suggests these attackers act more like an opportunistic cybercriminal, rather than an indomitable APT or nation-state. However, given the

Word	# Incidents	Word	# Incidents
document	89	sent	44
view	76	review	43
attach	56	share	37
click	55	account	36
sign	50	access	34

Table 5.5: Top 10 most common words across all 180 lateral phishing incidents.

arms-race and evolutionary nature of security, these lateral phishers could in the future increase the sophistication and potency of their attacks by drawing upon the account's prior emails to craft more targeted content.

Thematic Content (Lures): When labeling each phishing incident with a level of tailoring, we noticed that the phishing messages in our dataset overwhelmingly relied on one of two deceptive pretexts (lures): (1) an alarming message that asserts some problem with the recipient's account (and urges them to follow a link to remediate the issue); and (2) a message that notifies the recipient of a new / updated / shared document. For the latter 'document' lure, the nature and specificity of the document varied with the level of content tailoring. For example, whereas an attack with generic topic tailoring just mentioned a vague document, attacks that use enterprise-related tailoring switched the terminology to an invoice, purchase order, or some other generic but work-related document.

To characterize this behavior further, we computed the most frequently occurring words across our dataset's phishing messages. First, we selected one phishing email per incident, to prevent incidents with many identical emails from biasing (inflating) the popularity of their lures. Next, we normalized the text of each email: we removed auto-generated text (e.g., user signatures), lowercased all words, removed punctuation, and discarded all non-common English words; all of these can be done with open source libraries such as Talon [70] and NLTK [7]. Finally, we built a set of all words that occurred in any phishing email across our incidents and counted how many incidents each word appeared in.

Interestingly, our dataset's phishing messages draw on a relatively small pool of words: there are just 444 distinct, common English words across the texts of every phishing message in our dataset (i.e., every phishing email's text consists of an arrangement from this set of 444 words). In contrast, a random sample of 1,000 emails from our dataset contained a total of 2,516 distinct words, and only 176 of these emails consisted entirely of words from the phishing term set.

Beyond this small set of total words across lateral phishing emails, all but one incident contained at least one of the top 20 words, illustrating the reliance on the two major lures we identified. Figure B.7 in Appendix B.5 shows the full occurrence distribution of each word. Focusing on just the top ten words and the number incidents that use them (Table 5.5), the dominance of these two thematic lures becomes apparent. Words indicative of the "shared document" lure, such as 'document', 'view', 'attach', and 'review', each occur in over 23% of incidents, with the most popular

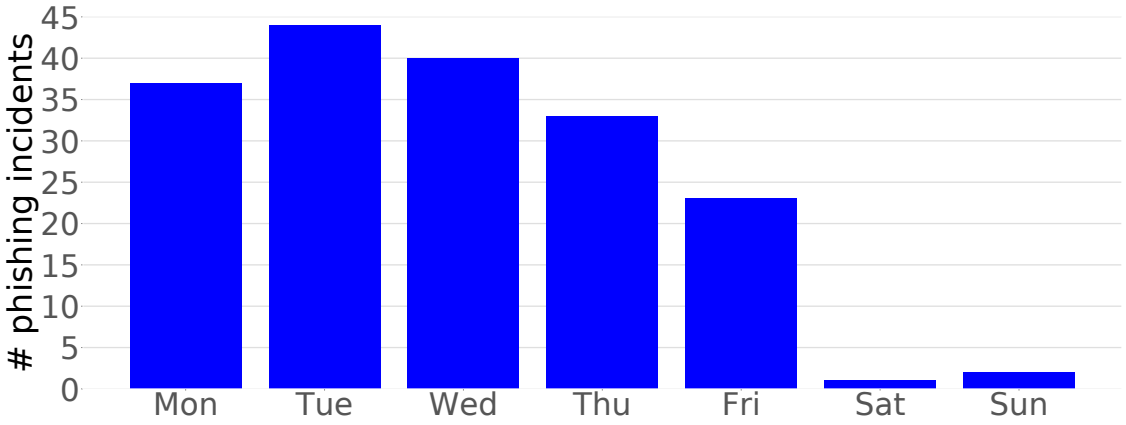

Figure 5.7: Number of lateral phishing incidents per day of week.

(document) occurring in nearly half of all incidents. Similarly, we also see many words from the account-related lure in the top ten: 'access', 'sign' (from 'sign on'), and 'account'.

Overall, while our dataset contains several instances of targeted phishing messages, the majority of the lateral phishing emails we observe rely on more mundane lures that an attacker can reuse across multiple organizations with little effort. The fact that we see this behavior recur across dozens of different organizations suggests either the emergence of a new, yet accessible, form of enterprise phishing and lateral movement, or an evolution in the way "ordinary" cybercriminals execute phishing attacks (expanding from external accounts that use clever spoofing to also leverage compromised, yet legitimate accounts).

Temporal Aspects of Lateral Phishing

Because attackers might not live or operate in the same geographic region as a hijacked account, prior work has suggested using features that capture unusual timing properties inherent in phishing emails [32, 38, 114]. Contrary to this intuition, in our dataset most lateral phishing attacks occur at "normal" times of the day and week. First, for 98% of lateral phishing incidents, the attacker sent the phishing email during a weekday. Additionally, the majority of attackers in our dataset send their phishing emails during the true account's normal working hours.

Day of the Week: As seen in Figure 5.7, all but three lateral phishing incidents occurred during a work day (Monday–Friday). This pattern suggests that attackers send their phishing emails on the same days when employees typically send their benign emails, and that the day of the week will provide an ineffective or weak detection signal. Moreover, 67% of incidents occur in the first half of the week (Mon–Wed), indicating that the lateral phishers in our dataset do not follow the folklore strategy where attackers favor launching their attacks on Friday (hoping to capitalize on reduced security team operations over the coming weekend) [100].

Time (Hour) of Day: In addition to operating during the usual work week, most attackers tend to send their lateral phishing emails during the typical working hours of their hijacked accounts.

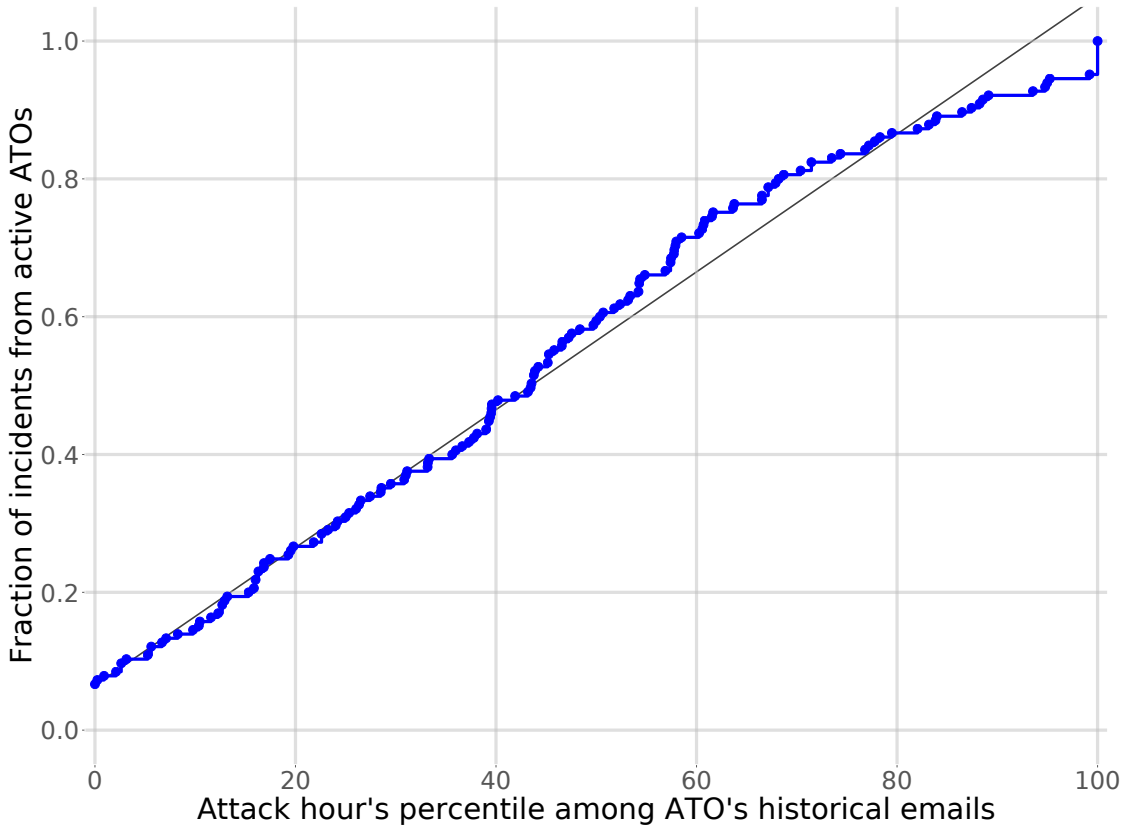

Figure 5.8: CDF of the fraction of incidents from active ATOs where the time (hour) of day fell within the x'th percentile of the hours at which the ATO's benign emails in the preceding 30 days were sent. Active ATOs are hijacked accounts that sent at least 1 non-phishing email within the 30 days preceding their lateral phishing email.

To assess the (ab)normality of an attack's sent-time, for each ATO, we gathered all of the emails that the account sent in the 30 days prior to their first lateral phishing email. We then mapped the sent-time of each of these historical (and presumably benign) emails to the hour-of-day on a 24 hour scale, thus forming a distribution of the typical hour-of-day in which each hijacked account usually sent their emails. Finally, for each lateral phishing incident, we computed the percentile for the phishing email's hour-of-day relative to the hour-of-day distribution for the ATO's historical emails. For example, phishing incidents with a percentile of 0 or 100 were sent at an earlier or later hour-of-day than any email that the true account's owner sent in the preceding 30 days.

Across all lateral phishing incidents sent by an active ATO, Figure 5.8 shows what hour-of-day percentile the phishing incident's first email occurred at, relative to the hijacked account's historical emails. Out of the 180 incidents, 15 incidents were sent by an "inactive" (quiescent) ATO that sent zero emails across all 30 days preceding their lateral phishing emails; Figure 5.8 excludes these incidents. Of the remaining 165 incidents sent by an active ATO, 18 incidents fall completely outside of the hijacked account's historical operating hours, which suggests that a

feature looking for emails sent at atypical times for a user could help detect these attacks. However, for the remaining 147 incidents, the phishing emails' hour-of-day evenly cover the full percentile range. As shown in Figure 5.8, the percentile distribution of phishing hours closely resembles the CDF of a uniformly random distribution (a straight $y = x$ line); i.e., the phishing email's hour-of-day appears to be randomly drawn from the true account's historical hour-of-day distribution. This result indicates that for the majority of incidents in our dataset (147 out of 180), the time of day when the ATO sent the attack will not provide a significant signal, since their sent-times mirror the timing distribution of the true user's historical email activity.

Thus, based on the attacks in our dataset, we find that two weak timing-related features exist: searching for quiescent accounts that suddenly begin to send suspicious emails (15 incidents), and searching for suspicious emails sent completely outside of an account's historically active time window (18 incidents). Beyond these two features and the small fraction of phishing attacks they reflect, neither the day of the week nor the time of day provide significant signals for detection.

Attacker Sophistication

Since most of our dataset's lateral phishers do not mine the hijacked account's mailbox to craft targeted messages, one might naturally conclude that these attackers are lazy or unsophisticated. However, in this subsection, we identify two kinds of sophisticated behavior that required some investment of additional time and manual effort: attackers who continually engage with their attack's recipients in an effort to increase the attack's success rate, and attackers who actively "clean up" traces of their phishing activity in an attempt to hide their presence from the account's legitimate owner. In contrast to the small number of attackers who invested time in crafting tailored phishing messages to a personalized set of recipients, nearly one-third (31%) of attackers engage in at least one of these two sophisticated behaviors.

Interaction with potential victims: Upon receiving a phishing message, some recipients naturally question the email's validity and send a reply asking for more information or assurances. While a lazy attacker might ignore these recipients' replies, 27 ATOs across 15 organizations actively engaged with these potential victims by sending follow-up messages assuring the victim of the phishing email's legitimacy. For example, at one organization, an attacker consistently sent brief follow-up messages such as "Yes I sent it to you" or "Yes, have you checked it yet?". In other cases, attackers replied with significantly more elaborate ruses: e.g., "Hi [Bob], its a document about [X]. It's safe to open. You can view it by logging in with your email address and password."

To find instances where a phisher actively followed-up with their attack's potential victims, we gathered all of the messages in every lateral phishing email thread and checked to see if the attacker ever received and responded to a recipient's reply (inquiry).[8] In total, we found that 107 ATOs received at least one reply from a recipient. Of these reply-receiving attackers, 27 ATOs (25%) sent a deceptive follow-up response to one or more of their recipients' inquiries.

[8]Office 365 includes a *ConversationID* field, and all emails in the same thread (the original email and all replies) get assigned the same ConversationID value.

Stealthiness: Separate from interacting with their potential victims, attackers might expend manual effort to hide their presence from the account's true owner by removing any traces of their phishing emails, particularly since lateral phishers appear to operate during the hijacked account's normal working hours. To estimate the number of these ATOs, we searched for whether any of the following emails ended up in the hijacked account's Trash folder, and were deleted within 30 seconds of being sent or received: any phishing emails, replies to phishing emails, or follow-up emails sent by the attacker. The 30 second threshold distinguishes stealthy behavior from deletion resulting from remediation of the compromised account. In total, 30 attackers across 16 organizations engage in this kind of evasive clean-up behavior.

Of the 27 ATOs who interactively responded to inquiries about their attack, only 9 also exhibited this stealthy clean-up behavior. Thus, counting the number of attackers across both sets, 48 ATOs engaged in at least one of these behaviors.

The sizeable fraction of attackers who engage in a sophisticated behavior creates a more complex picture of the attacks in our dataset. Given that these attackers do invest dedicated and (often) manual effort in enhancing the success of their attacks, why do so many of them (over 90% in our dataset) use non-targeted phishing content and target dozens to hundreds of recipients? One plausible reason for this generic behavior is that the simple methods they currently use work well enough under their economic model: investing additional time to develop more tailored phishing emails just does not provide enough economic value. Another reason might be that growth of lateral phishing attacks reflects an evolution in the space of phishing, where previously "simple" external phishers have moved to sending their attacks via lateral phishing because attacks from (spoofed) external accounts have become too difficult, due to user awareness and/or better technical mitigations against external phishing. Similarly, lateral phishing might represent a simple and low-cost way for opportunistic attackers to conduct lateral movement through an organization, given limited access and potentially limited skills. Ultimately, based on our work's dataset, we cannot soundly answer why so many lateral phishers employ simple attacks, and leave it as an interesting question for future work to explore.

5.7 Chapter Summary

In this chapter, we explored the nature of lateral phishing attacks across more than 100 million employee-sent emails from 92 enterprise organizations. We also developed and evaluated a new detector that found many known lateral phishing attacks, as well as dozens of unreported attacks, while generating a low volume of false positives. Through a detailed analysis of the attacks in our dataset, we uncovered a number of important findings that inform our mental models of the threats enterprises face, and illuminate directions for future defenses. Based on our dataset, we estimate that 14% of our randomly sampled organizations, ranging from small to large, experienced lateral phishing attacks within a seven-month time period, and that attackers succeeded in compromising new accounts at least 11% of the time. While some attackers send phishing emails with tailored message content to a targeted set of recipients, most attackers in our dataset favor strategies that employ non-personalized phishing attacks that can be readily used across different organizations.

Despite this apparent lack of sophistication in tailoring and targeting their attacks, 31% of our dataset's lateral phishers engaged in some form of sophisticated behavior designed to increase their success rate or mask their presence from the hijacked account's true owner. Additionally, over 80% of attacks occurred during a typical working day and hour, relative to the legitimate account's historical emailing behavior; this suggests that these attackers either reside within a similar timezone as the accounts they hijack or make a concerted effort to operate during their victim's normal hours. Ultimately, the findings of this chapter illustrate that many organizations encounter attackers who leverage compromised enterprise accounts to spread laterally. Although the attacks we observe show varying levels of targeting and sophistication, the magnitude of victim organizations and the apparent success of these attacks highlights the value and need to hunt for malicious activity across an enterprise's internal accounts and environment.

Chapter 6

Modeling and Detecting Lateral Movement Attacks

6.1 Introduction

Many enterprise attacks, ranging from stealing sensitive data to damaging critical infrastructure, often require adversaries to move beyond their initial point of compromise to achieve their goal [67, 78, 120]. For example, an employee compromised by a spearphishing attack often does not have all of an organization's sensitive secrets readily stored on their machine; thus, attackers typically need to move onto other machines in an enterprise's network that contain their desired data. The set of internal movements that such attackers make is known as *lateral movement* [27, 119].

In the preceding chapter, we characterized and developed a detection strategy for one form of lateral movement: lateral phishing attacks. Whereas lateral phishing attacks primarily enable an attacker to move from one cloud *account* to another, in this chapter, we focus on detecting lateral movement activity where an attacker moves from one compromised enterprise *machine* to other internal machines within an organization. This machine-to-machine spreading embodies what the security community has traditionally defined as enterprise lateral movement; based on public breach reports, this definition reflects the predominant way that sophisticated attackers expand their internal access in a compromised organization [15, 27, 79, 119]. Accordingly, for the remainder of this chapter, we simply refer to this type of malicious machine-to-machine movement as "lateral movement".

Prior work on detecting lateral movement has typically either required kernel-level instrumentation at each host in the network [44,50,75], which is difficult to deploy at scale, or used traditional anomaly detection methods, which suffer from an impractically high false alarm rate [59, 68, 69, 111]. The work we present in this chapter addresses these limitations by developing new methods for detection at scale, using logging data already captured on typical enterprise networks, and with a manageable rate of false alarms.

We present Hopper, a practical system for detecting lateral movement attacks. Hopper builds a graph of user movements (logins) between internal machines and then identifies suspicious move-

ment paths within this graph using novel algorithms. Hopper addresses three key challenges encountered in prior work: capturing the full context of a user's movement over time using commonplace logs that only report point-wise login activity; detecting lateral movement attacks amidst a vast volume of benign events with only a few or no labeled attack instances; and maintaining a low false positive rate.

To overcome these challenges, Hopper employs a simple observation: attacks generally perform lateral movement in order to access a machine that the attack's initial victim lacks access to. Thus, as part of their lateral movement activity, attackers will need to acquire and switch to a new set of credentials that enables the sought-for access. Leveraging this observation, Hopper traces the paths of logins that each actor (enterprise user) makes and detects lateral movement by identifying paths for which: (1) the path's actor switches credentials, and (2) the path performs a login into a destination that the path's actor should not access (Section 6.4). To help us identify these types of attacks, we develop new methods for correlating point-wise activity over time and detecting rare attacks in the absence of labeled data.

To manage false positives, Hopper uses a new specification-based anomaly detection algorithm to rank and uncover potential lateral movement activity. Traditionally, intrusion detection research has often relied on either signature-based detection or anomaly detection. Signature-based methods have seen wide deployment, but they require substantial manual effort to craft, and are often overly specific and weak at detecting new attacks. In principle, anomaly detection has the potential to detect novel attacks with less effort from analysts, but in practice it tends to suffer from an intractable false positive rate, typically because of anomalous-but-benign behavior: situations that are unusual among historical data, yet are benign (e.g., configuration errors, new software versions, distribution shift). This occurs because anomaly detection identifies events that are unusual among benign traffic, but it has no knowledge of what is suspicious: which factors indicate a possible attack, as opposed to some other kind of anomalous-but-benign behavior.

Instead of narrow signatures, where each signature is narrowly tailored to a single attack method, we formulate a specification that captures fundamental characteristics of any successful attack. Reflecting the key observations we made above, our specification states that any successful lateral movement attack will (1) switch to a new set of credentials and (2) eventually access a server that the original actor could not access. To reduce the rate of false positives, we combine this specification with anomaly detection. In particular, we raise an alert on any event that is both identified as unusual by our anomaly detector and identified as suspicious by the specification. This hybrid approach allows Hopper to achieve the best of both worlds and detect lateral movement with a modest false positive rate.

This chapter describes our approach in more depth. First, we present a method for generating a set of login paths from commonly collected enterprise authentication logs (Section 6.5). Next, we introduce a new detection algorithm that enables Hopper to identify particularly suspicious login paths while generating a tractable volume of alerts (Section 6.6). As part of this detection algorithm, we develop an anomaly scoring technique that enables Hopper to rank the relative suspiciousness of login paths without the use of any labeled data. Finally, we evaluate Hopper on a 15-month enterprise dataset that contains over 780 million internal login events (Section 6.7). This dataset includes one real lateral movement attack performed by a professional red team and a set

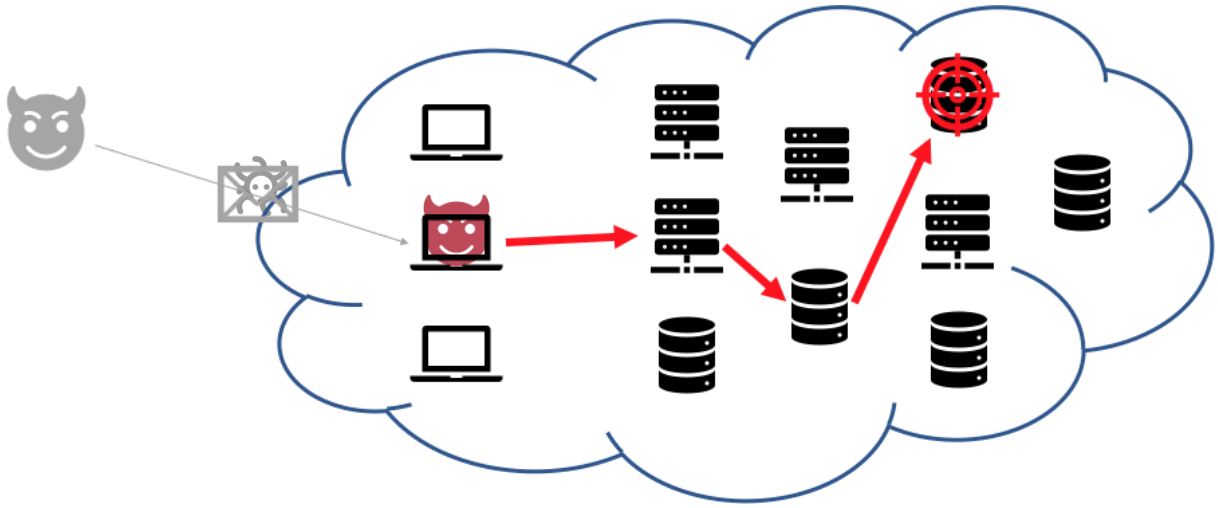

Figure 6.1: Lateral movement, depicted as red arrows, is the set of movements that an attacker makes between *internal* machines within an enterprise to access their desired machine(s) and data.

of 326 simulated attacks that spans a diverse array of real-world attacks (ranging from ransomware to stealthy, targeted machine compromise). On this real-world dataset, Hopper can detect 95.4% attacks (312 / 326) while generating an average of 9 alerts per day: more than an 8× improvement (fewer alerts) compared to the best-performing prior work [111].

6.2 Background

Attackers frequently need to spread beyond the initial "foothold" machine they have compromised onto other machines, in search of the servers and necessary credentials that provide them with their desired data and capabilities. *Lateral movement* encompasses the internal movements an attacker makes among machines *within* an enterprise to achieve their goal.

Security Model

Below, we describe the security goals and types of lateral movement that our detection system, Hopper aims to uncover.

Detection Goals: We design Hopper with four goals in mind:

1. Only relies on commonly-collected enterprise authentication logs.
2. Generates a very low volume of false positives, tenable for sites to analyze.
3. Detects a diverse range of lateral movement attacks.
4. Operates effectively given no labeled attack instances.

Nodes (Source + Destination Machines)	Edge (Login)
Hostname	Timestamp
Client vs. server	Target username
Owner's username (clients only)	

Table 6.1: The information contained for each login event in our data. Hopper uses this information to construct a graph of user movement (Section 6.4), where each login creates a unique edge between two nodes (internal machines) in the login graph.

We consider Hopper successful if it produces an alert for any login made by attacker conducting lateral movement. Upon confirming the presence of an attack, an organization's security team can use forensic techniques from prior work [45, 51, 129] to perform additional analysis and remediation.

We focus on developing detection in a setting where an organization has a team of security analysts with a fixed time budget for reviewing alerts. In particular, we design Hopper to score a set of movement paths in terms of how problematic the activity appears to be, allowing an organization to specify their own bound on the number of alerts (most suspicious paths) that Hopper generates. Based on prior work [9, 49] and the practical experiences of our industry collaborators, this alert-budget design accurately reflects a real-world operating model for many organizations.

Hopper aims to detect many prevalent types of real-world lateral movement, while generating a tractable number of false positives. Balancing these goals, Hopper might not detect novel cases of lateral movement proposed in previous work, such as attacks where the adversary moves strictly via hijacking or piggybacking on top of a sequence of logins made by legitimate users [80] and attacks where an adversary successfully "poisons" an organization's login dataset with historical logins that make their final attack path appear benign and frequently traveled.

Threat Model: Similar to prior work, we focus on detecting interactive and credential-based lateral movement attacks [111]. Under this threat model, we assume that an attacker has managed to compromise an initial "foothold" machine within the enterprise, but they (1) subsequently need to acquire additional credential(s) to access the data or systems they ultimately seek, and (2) move between machines via login or remote command execution events that use a set of credentials for authentication, rather than by accessing machines by exploiting vulnerabilities. Additionally, this threat model focuses on attackers who manually perform each of the movement (login) operations during their attack, as opposed to an attack that installs malware that moves to new systems autonomously.

The threat model we outline reflects the behavior of many real-world lateral movement attacks, ranging from targeted nation-state attacks [15, 46, 71, 79, 82, 93, 116] to newer, stealthier forms of ransomware attacks [39, 120]. Our threat model does not consider attackers whose initial attack already gains them their desired access since such attacks lack any lateral movement. We note, however, that organizations can employ techniques developed by complementary prior work [30, 36, 42, 104] to restrict the users and machines that have direct access to sensitive data or powerful functionality, thereby reducing the prospect of such attacks.

6.3 Data

Our work uses a collection of successful login events between internal machines at Dropbox, a large enterprise that provides cloud collaboration services to hundreds of millions of users. Whenever a machine receives a remote access attempt from another machine (e.g., an inbound ssh session or a remote command execution issued via utilities like psexec), the receiving machine generates a record of a remote "login" attempt. Because all modern operating systems record these login events by default, most organizations collect these logs as part of standard security best practices.

This data provides visibility into the internal logins between machines within Dropbox's corporate network, such as client laptops, authentication servers (e.g., Windows Domain Controller), and a variety of infrastructure and application servers (e.g., DNS servers, machines that test and build applications, and analytics servers). Representative of the heterogeneous nature of modern enterprises, the logins in our data span a variety of authentication protocols (e.g., Kerberos, NTLM, and ssh) across many types of devices (laptops, physical servers, and virtual machines) and operating systems (Windows, Mac OSX, and Linux).

Data Size and Schema

Our data consists of 784,459,506 successful login events from Jan 1, 2019 to Apr 1, 2020 (15 months). As shown in Table 6.1, each login event contains the time of the login, the target username of the login, the source and destination machines that initiate / receive the login, and a few additional properties about these machines.

Despite this large volume of logins we found that the vast majority of logins did not reflect meaningful remote access events (i.e., did not enable a user to remotely execute commands or access sensitive data on the destination machine). Instead, many of these login events reflect artifacts produced by enterprise logging and/or automated processes, which prior work has also observed [59]. For example, most Windows logins correspond to uninteresting authentication records generated by an enterprise's Domain Controllers during Kerberos-based authentication, and by networking printers that record user "logins" when receiving print job requests. Other examples of spurious login events include remote authentication into an enterprise's update and logging servers that machines perform to transmit their host logs or receive updates, and logins performed by automated scripts that use restricted credentials to perform limited operations. Hopper applies four filtering rules to remove these logins from our dataset, as described in Appendix C.1. Excluding these spurious logins, our dataset contains 3,527,844 successful logins, with a median of 4,098 logins per day. These logins span 634 accounts and occur between 2,327 machines.

Ethics

This work involved a collaboration between academia and industry. Our research used an existing, historical dataset of employee logins between internal machines at Dropbox, which enterprises commonly collect to mitigate attacks and secure their internal environment. Only authorized security employees at Dropbox accessed this data; no sensitive data or personally identifying infor-

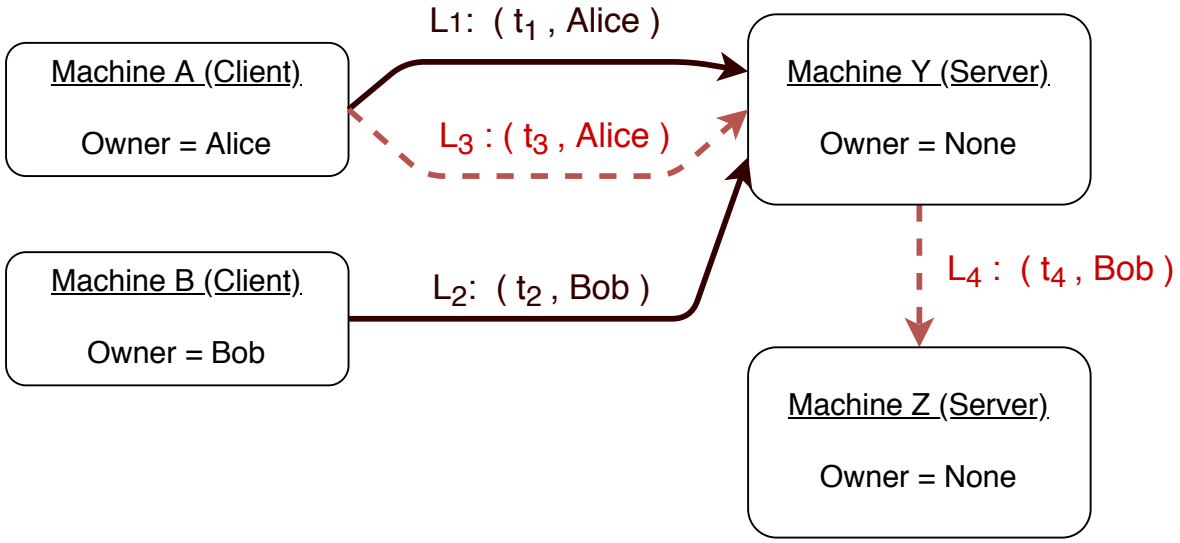

Figure 6.2: An example of a simple login graph. Solid black edges (L_1 and L_2) correspond to benign login events. Dashed red edges (L_3 and L_4) correspond to a lateral movement attack path. Each node in the graph represents an internal enterprise machine (e.g., Machine A, B, Y, Z), and contains the information in Column 1 of Table 6.1. Each edge in the graph corresponds to a unique login event and contains the information in Column 2 of Table 6.1.

mation was shared outside of Dropbox. Additionally, the machines that store and operate directly on data from Dropbox's customers reside on separate, segmented infrastructure; our study did not involve that infrastructure or access any customer-related data. This project underwent internal review and received approval by the legal, privacy, and security teams at Dropbox.

6.4 Hopper: System Overview

We begin this section with an overview for how our system, Hopper, identifies suspicious login paths. Next, we describe the graph that Hopper constructs from logs of remote logins, discuss the challenges encountered when applying a standard anomaly detection approach on our dataset, and summarize Hopper's architecture.

Our Approach: Hopper constructs a graph of user logins between internal machines and then detects lateral movement by identifying "suspicious" paths in this graph. A suspicious path corresponds to a sequence of logins made by a single actor with two properties: (1) the path has at least one login where the actor uses a set of credentials that does not match their own, (2) the path accesses at least one machine that the actor does not have access to under their own credentials.

Motivating Intuition: This approach leverages a simple yet powerful observation: in many real-world enterprise attacks, adversaries conduct lateral movement to acquire additional credentials and access new machines that their initial foothold did not have access to [28,46,71,79,82,93,116].

For example, at many organizations, access to sensitive data and/or powerful internal capabilities requires a special set of credentials and privileges, which most enterprise users lack. Thus, attacker lateral movement will produce paths that use a new set of credentials (property 1) and access machines that their initial victim could not access (property 2). Moreover, these suspicious characteristics also correspond to movement that we do not expect benign paths to exhibit: users should access machines under their own credentials and they should only login to machines that they have legitimate privileges to access. Each of our attack properties correspond to violations of these expected behaviors. We show that this approach yields orders-of-magnitude fewer alerts than a traditional anomaly detection approach, such those proposed in prior work (Section 6.7).

The Login Graph

Given a set of logins, Hopper constructs a directed multi-graph that captures the interactions among users and machines within an enterprise. Figure 6.2 shows a simple example of a login graph constructed by Hopper. Each login creates a directed edge in the graph, where the edge's source and destination nodes correspond to the machine initiating and receiving the login, respectively. Edges represent unique, timestamped logins from the source to the destination machine; multiple logins between the same two machines generate multiple edges. Each edge is annotated with a target username, which is the account that was logged into on the destination machine; the target username specifies what user and permissions the login's session will operate under on the destination machine. Table 6.1 shows the information contained in our login data used to label each edge and node in the graph.

Login Paths and Causal Users: A path of logins corresponds to a series of connected edges, where each edge is "caused" by the same actor: i.e., a path is a sequence of connected logins that one actor made from the path's starting machine to the path's final destination. We use the term *causal user* to refer to the actor whose machine initiated a path of logins, which might not be the same as the *target user* recorded in each login. The causal user is the original actor responsible for making these logins (taken from the first edge in the path), while each login's target user reflects the credentials that the login's destination machine received.

For example, in Figure 6.2, an attacker compromises Alice's corporate machine (A) and makes a series of internal logins that forms a two-hop lateral movement path from Machine A to Machine Z. The attacker first logs into Alice's account on Y using the initial victim's credentials, shown as L_3. Then the attacker compromises Bob's credentials on Machine Y and uses them to log into Bob's account on Z, labeled L_4. For each of the logins in this attack path, Alice is the causal user, since all of the logins were made (caused) by a user starting from Alice's machine.

Challenge: Anomalies at Scale

Prior work detects lateral movement by identifying logins that traverse rare graph edges, under the assumption that attacker movement will occur between users and/or machines that rarely interact with each other [8, 69, 111]. While intuitive, these approaches ultimately generate too many false

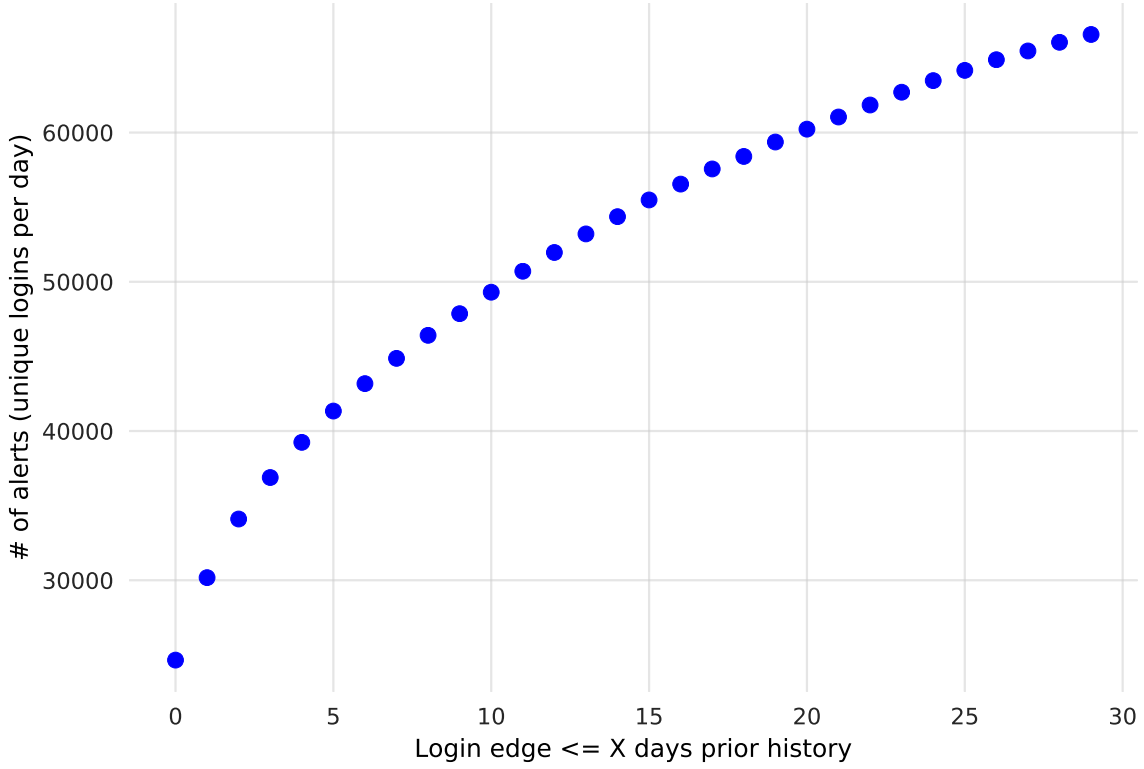

Figure 6.3: The number of logins with rare graph edges, which have occurred in $\leq$ X days in the 60 days preceding the login. The graph shows the number of alerts (de-duplicated to one unique edge per day) that a simple anomalous login detector would generate.

positives for practical use, due to the diverse array of rare-but-benign behavior that occurs within large-scale enterprises. Even after applying the steps that Hopper takes to eliminate artifacts and automation (Section 6.3), we found that tens of thousands of logins create "rare" graph edges in our dataset. Figure 6.3 shows the number of logins that a simple anomaly detector would flag if it alerted on any login whose edge (source machine, destination machine, and target username) has occurred $\leq X$ days in the past 60 days of prior logins, after de-duplicating them to only produce one alert per unique edge each day. Even at the most aggressive threshold of 0 prior days, i.e., edges that have never occurred in recent history, this anomaly detection approach would still produce over 24,000 alerts across our dataset (over 1,600 alerts per month).

Analyzing a random sample of these extremely-rare-edge logins, we found many reasons for these benign anomalies. For example, some of these logins correspond to users logging into machines they rarely access to perform maintenance (e.g., either system administrators or regular users serving on their team's on-call rotation). In other instances, these logins reflect activity by new users or employees returning from a long vacation, as well as users simply accessing new or rare-for-their-role services. Moreover, selecting such an aggressive threshold will cause this simple anomaly detector to miss many attacks, particularly ones that exhibit some level of stealthiness,

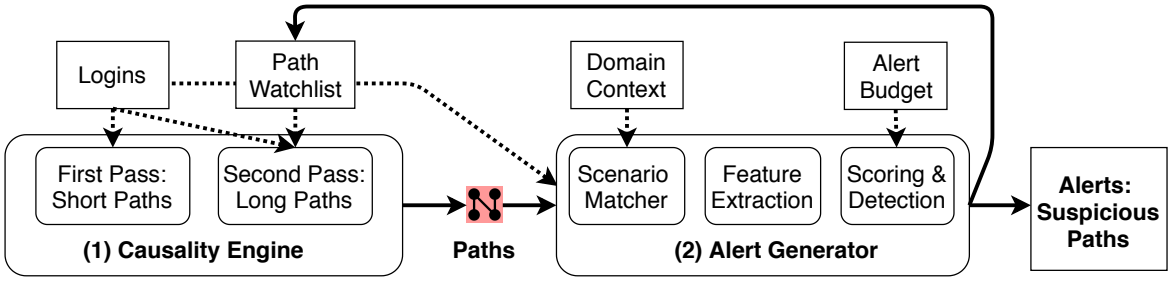

Figure 6.4: Hopper analyzes login events between internal machines within an enterprise and generates alerts for paths of logins that correspond to suspicious lateral movement activity. Hopper has two key components: (1) a causality engine that infers a set of causal paths that a login might belong to (Section 6.5), and (2) detection and scoring algorithms that decide whether to alert on a login path (Section 6.6).

as shown in Section 6.7. Although previous work introduces techniques to refine this anomaly detection approach, they still produce too many false positives (Section 6.7). By re-framing the definition of an attack path from simply paths that rarely occur to paths that contain the two key characteristics we highlight, Hopper can significantly reduce the volume of false positives it produces while still detecting a range of real-world lateral movement attacks.

System Architecture

Hopper consists of two stages, shown in Figure 6.4. The first stage of Hopper (Section 6.5) consists of a "causality engine" that produces a set of login paths from individual login events. This stage allows Hopper to coarsely infer the causal information for each login and the broader path it belongs to. The second stage of Hopper (Section 6.6) takes a set of login paths and decides whether to generate an alert by identifying whether any of these login paths contain the two key attack properties described above. During this final stage, Hopper prunes common benign movement paths, extracts a set of attack-driven features for each path, and uses a combination of detection rules and a new anomaly scoring algorithm to compute the "suspiciousness" of each login path. In some cases, a path will contain enough suspicious characteristics that Hopper can use a simple rule-set to generate an alert for the path. In other scenarios, Hopper's causality engine does not provide enough clarity to directly label a path as malicious versus benign. For this latter case, Hopper takes a user-specified alert budget that bounds the number of alerts it produces for these unclear scenarios. Using this alert budget and the suspiciousness scores generated by its anomaly scoring algorithm, Hopper outputs a list of the most suspicious paths, bounded in length by the specified alert budget.

Path Component	Description
Focal Hop	The path's final hop, or the path's credential-switching hop (Section 6.5)
Hop List	List of logins in the path (Table 6.1)
Causal User	Username of the employee whose machine initiated the path
Path Likelihood	Fraction of the # of causal paths that Hopper inferred for the focal hop

Table 6.2: Information contained within an inferred causal path generated by Hopper's causality engine (Section 6.5). Given a new login, Hopper infers a set of these causal paths, each of which reflects a sequence of logins that an actor could have made up to and including the new login (the path's focal hop). In the case of an extended path from Hopper's watchlist, the focal hop corresponds to the credential-switching login that placed the path on the watchlist.

6.5 Generating Causal Login Paths

Although standard machine authentication logs contain a rich set of information, they reflect point-wise activity that lacks context about the broader user activity surrounding a login, such as from who and where the login originated. For example, in Figure 6.2, given login L_4 in isolation, a detector lacks insight into whether *Bob* accurately reflects the user responsible for making the login, or whether another user such as Alice has stolen Bob's credentials and is using them in a malicious login. Thus, for each login (L_i) that occurs, the first stage of Hopper runs a "causality engine" that coarsely infers the broader path of movement that a login belongs to and the "causal" user responsible for initiating the movement path. To do so, Hopper uses a time-based heuristic to infer a set of "causal paths" for L_i, where each path produced Hopper corresponds to a unique sequence of connected logins that includes L_i and occurred within the maximum time limit for a login session.

In some cases, Hopper can cleanly infer a single path and causal user for a new login, allowing Hopper classify the login path using a simple set of detection rules described in Section 6.6. However, some logins will cause Hopper to generate many inferred paths with different causal users, creating uncertainty about the true causal user and path for L_i. To handle this latter case, Hopper relies on an anomaly scoring algorithm when deciding whether to alert on an inferred path (Section 6.6). In the remainder of this section, we describe how Hopper's causality engine infers a set of paths for a new login, and defer the details about detection and alerting to Section 6.6.

Inferring Causal Paths

Each of the causal paths that Hopper infers for L_i contains the information in Table 6.2: the path's focal hop (the login, L_i, that Hopper constructed the path for), a list of all of the logins (hops) in the inferred path, the path's causal user, and the path's "likelihood", which quantifies Hopper's

certainty about whether the current path reflects the true causal path of L_i; Hopper estimates this likelihood as a simple fraction: 1 / (the # of causal paths that Hopper inferred for the focal hop).

Identifying Causally-Related Logins: Hopper produces a set of causal paths by running a backwards-tracing search from L_i to identify a sequence of causally-related logins that include L_i. Two logins are causally related if they (1) form a connected set of edges in the login graph and (2) occur within T hours of each other. More formally, we say that L_k is a causal, inbound login for L_i if the destination of L_k equals the source machine of L_i, and L_k occurred within 24 hours prior to the time of L_i. We choose a threshold of 24 hours as a proxy for the maximum duration of a login session at Dropbox; through machine and network configurations, sessions that exceed this duration require the source machine to re-authenticate, which produces a fresh login event in our data set. For example, in Figure 6.2, L_1, L_2, and L_3 are all causal hops for L_4 if they occurred within 24 hours prior to t_4.

Using this causal rule, Hopper can infer a set of login paths by identifying all of the causal logins for L_i, and then recursively repeating this search on each of those causal logins. This process is similar to provenance and taint-tracking methods that trace the flow of information from a sink (L_i's destination machine) back to its source (e.g., a client machine at the root of L_i's login path) [44, 50, 51]. As with these prior flow-tracking methods, naive backwards-tracing creates a "dependency explosion", where each backwards step can exponentially increase the number of (potential) causal paths that Hopper infers yet only one of these inferred paths corresponds to L_i's true causal path. To address this problem, Hopper's causality engine uses a two-pass algorithm that only generates long paths (three or more hops) for logins that could plausibly belong to an attack path.

Hopper's causality engine takes three inputs: a new login event (L_i) to infer paths for, a set of all recent logins, and an incrementally updated "watchlist" of suspicious login paths. The first pass of Hopper's causality engine aims to make a greedy decision about L_i by producing a set of one-hop or two-hop causal paths, and then passes these paths to its alert generator (Section 6.6). If Hopper's alert generator cannot make a clean benign versus malicious decision, the causality engine's second pass continues tracking the path and continually resubmits an updated, longer-length path to the alert generator for each future login in the path.

First Pass: Constructing Short Causal Paths

Given a login, the first pass of Hopper's causality engine produces either a one-hop path (single login) or a set of two-hops paths, depending on whether the login occurs from a client or server. Let L_i represent a login from source machine Y to destination machine Z with a target user of Bob.

If Y is a server, Hopper produces a set of two-hop paths by running a single iteration of backwards-tracing to infer a set of inbound causal logins for L_i. Hopper then pairs each of these inbound logins with L_i to form a set of two-hop paths that each contain the information in Table 6.2. For each path, Hopper sets L_i as the focal hop and adds the two logins to the path's hop list. Hopper sets the path's causal user equal to the target username of the inbound hop. The Path's likelihood equals 1 divided by the total number of two-hop paths that Hopper inferred for L_i. For

example, in Figure 6.2, assuming that L_1, L_2, and L_3 occurred within 24 hours prior to L_4, Hopper will produce 3 causal paths for L_4. Each of these paths will list L_4 as their focal hop and have a "Path Likelihood" of $1/3$. Both the attack path (L_3 to L_4) and the path from L_1 to L_4 will list Alice as their causal user, and the inferred path from L_2 to L_4 will list *Bob* as its causal user.

In contrast, if Y is a client, Hopper does not perform any backwards-tracing. Instead, Hopper's causality engine outputs a single one-hop path with a hop list and focal hop equal to L_i, and a "Path Likelihood" of 1. For the path's causal user, Hopper takes the owner of Y and treats that username as the causal user: clients typically correspond to the start of a user's movement path and logins from these machines should use the credentials of their owner. Because the overarching goal of Hopper's causality engine is to infer the causal user and corresponding path of a new login, Hopper does not need to perform any backwards-tracing inference since client logins mark the start of a path and these machines already contain the information about the login's causal user (i.e., the client's owner). In the example from Figure 6.2, Hopper generates three one-hop paths: one for each of L_1, L_2, and L_3.

Second Pass: Tracking Long Causal Paths

The second pass of Hopper's causality engine allows Hopper to trace and construct paths of arbitrary length (three or more hops). To do so, Hopper maintains a watchlist of earlier paths that Hopper's detection algorithm could not clearly label as benign or suspicious. Initially, this watchlist starts off empty. As Hopper processes and infers causal paths for new logins, it incrementally adds some of these paths to the watchlist if Hopper's alert generator cannot output a binary label for the path. The second pass of Hopper's causality engine tracks the paths on this watchlist and checks whether a new login extends any of these paths; if so, Hopper resubmits each of these newly extended paths for reclassification.

Extending a watchlist path: For every new login (L_i) that occurs, Hopper runs both passes of its causality engine. During this second pass, Hopper runs its path inference algorithm to identify all of the watchlist paths that L_i extends; i.e., where Hopper infers that the final hop of a watchlist path is a causal, inbound login for L_i. If L_i extends any watchlist path, Hopper creates a new "extended path" (P_2) by copying the contents of the watchlist path and then appending L_i to P_2's "Hop List" (all of the other path attributes remain the same as in the original watchlist path). Hopper then takes P_2 and submits it to its alert generation algorithm, which outputs either a binary verdict (benign versus suspicious) or adds P_2 to the watchlist.

Pruning the watchlist: As Hopper iterates over the watchlist paths during this second pass, it removes any path where the final hop occurred over T hours prior to L_i, where T reflects the maximum duration of a remote session (e.g., 24 hours). Because the time between path's last login and L_i exceeds the maximum session duration, the watchlist path could not have caused L_i unless the causal user performed a session reconnect or fresh login; both of these cases will generate a new login event (and corresponding paths) in our data.

Causality Engine's Coverage: This two-pass approach enables Hopper to efficiently trace multi-hop attack paths and identify the logins that involve a (potentially) suspicious use of credentials.

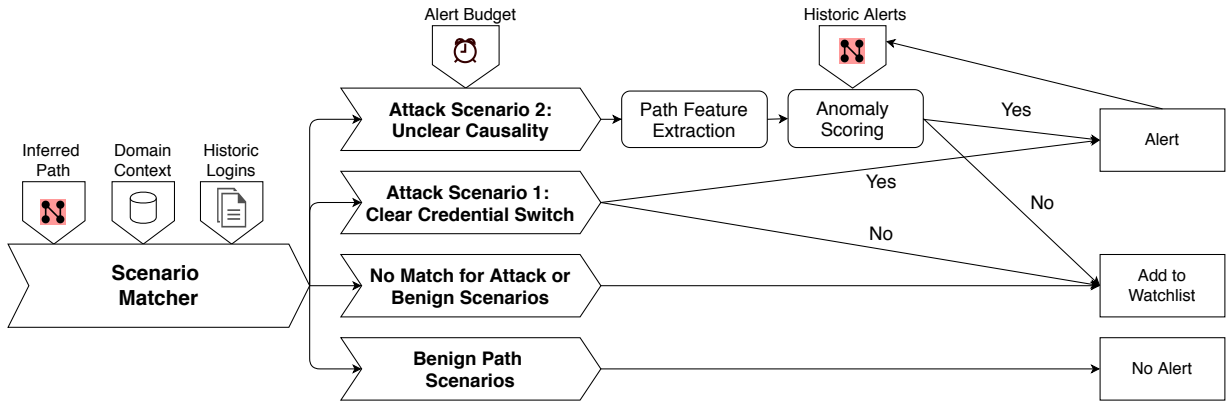

Figure 6.5: Architecture of Hopper's alert generator (Section 6.6). Given a login path inferred by its causality engine (Section 6.5), Hopper checks whether the path matches a benign scenario, an attack scenario, or none of these scenarios. Based on the path's scenario, Hopper either discards the path, generates an alert if the scenario's detector triggers, or adds the path to its watchlist.

Any path that involves a switch in credentials will contain at least one login that exhibits this switch, which the first pass of Hopper's causality engine will identify. Each time an attacker switches to a new set of credentials, Hopper will generate an additional set of causal paths with a new focal hop that reflects the credential-switching login. For any of these paths, if the attacker does not immediately access a new destination during this credential-switching login, then Hopper's detection algorithms (described below) will add the path onto the watchlist. This step allows the causality engine's second pass to continue tracking these potentially dangerous paths and the destinations they access in the future.

6.6 Detection and Alerting

For each login path, Hopper's alert generation stage decides whether the path is benign or suspicious, and takes one of three actions: generate an alert, discard the path, or add the path to the causality engine's watchlist (Section 6.5).

Hopper makes this decision given three inputs: a login path to classify (Table 6.2), a set of historical login events for feature extraction, and a user-provided "budget" that controls how many daily alerts Hopper should produce. Given these inputs, Hopper matches a path to a movement scenario: one of four benign scenarios, one of two attack scenarios, or none of these attack and benign scenarios. Paths that match a benign scenario do not generate an alert. For the remaining paths, Hopper matches the path to an attack scenario and applies the scenario's corresponding detector, which extracts a set of features for the path and then applies either a rule set or an anomaly scoring algorithm to determine whether the path is suspicious, producing an alert if so. If a path does not match any benign scenario and does not generate an alert under either detector, then Hopper adds the path to its watchlist for further tracking.

Benign Movement Scenarios: Hopper first attempts to identify if a new path corresponds to one of four benign movement scenarios. In the first benign scenario, Hopper marks a path as benign if every one of its logins uses its causal user's credential (i.e., the logins' target username always matches the causal username); because these paths do not switch credentials, Hopper discards them. As described in Appendix C.2, Hopper also labels a path as benign if it corresponds to one of three other benign and low-risk scenarios: one-hop paths from new machines or machines undergoing re-provisioning for a new owner; one-hop paths that use a service account with limited permissions; and paths that visit a (domain-provided) set of hardened machines that do not enable credential switching.

Attack Scenarios: If a path does not match any of these benign scenarios, then Hopper determines whether it matches the following two attack scenarios; if so, applies the appropriate detector to see whether it should produce an alert. First, if the path contains a login that switches credentials and the causality engine gives us high confidence in the inferred path, Hopper applies a simple specification-based rule set to classify the path as suspicious or not (Section 6.6); we call this a "clear credential switch" path. However, because of imperfect information, Hopper does not always have high confidence in the inferences drawn by its causality engine. For instance, in Figure 6.2, if L_2 and L_3 both occurred shortly before L_4, Hopper's causality engine won't be sure whether it should causally associate L_4 with L_2 or L_3, so it will output both possible paths. Although the path L_3 to L_4 contains a credential switch, the other path (L_2 to L_4) does not, so Hopper cannot be sure whether this represents a true credential shift or simply imprecision in the causality inference. Because of the uncertainty about these inferred paths, the second detector evaluates how suspicious and anomalous each such path is with a probabilistic scoring algorithm and alerts if the path has one of the most suspicious scores in recent history.

Attack Scenario 1: Paths with a Clear Credential Switch

Paths with a clear credential switch contain at least one login whose target username does not match the causal user that Hopper inferred for the path (the first key attack property). For these paths, Hopper generates an alert if the path accesses any destination that its causal user has never accessed in prior history (in the historical "training" logins provided to the alert generator); otherwise it adds the path to the watchlist.

More formally, let P represent an inferred path with a causal user of Alice and Dest_P refer to the destination machines across all of the login hops in P. Additionally, let AllUsers_P equal the set of causal users across every path that Hopper's causality engine produced for P's focal hop. Hopper generates an alert if P contains the two key attack properties:

1. Attack Property 1: P contains at least one login L_i with target user TargetUser_i, where $\text{TargetUser}_i \neq$ Alice and $\text{TargetUser}_i \notin \text{AllUsers}_P$.

2. Attack Property 2: P contains at least one destination in Dest_P (e.g., Machine Z) where Alice has never logged into Z in the historical training data (e.g., past 30 days).

Intuitively, these detection rules capture the two suspicious properties of a lateral movement path: an attacker conducts lateral movement to access a machine that their initial victim lacked access to (the second rule). To access this new machine, the attacker acquired and used a new set of credentials with the necessary permissions (the first rule, which captures the switch from the attacker's initial credentials to the new credentials).

This scenario captures two types of attack paths: an attacker who uses a new set of credentials from a client and an attacker who uses new credentials from a server that the credential's legitimate user has *not* recently accessed. Attackers might acquire new credentials through a variety of means, such as lateral movement onto various servers, internal phishing, or by mining internal documentation and wikis. Attackers can then use these new credentials to access their target machines from either their initial victim's device (a client, such as a laptop), or from some other machine.

If the attacker uses the new credentials from a client, the movement will be detected. Hopper's causality engine will produce a one-hop path for each malicious login from this client machine (Section 6.5). These one-hop paths contain a single causal user, the client's owner, and AllUsers_P contains only this owner. Because the attacker accesses target machines that the original owner could not access via a new set of credentials, these one-hop paths will have a clear credential switch from the causal user and thus will be detected.

If the attacker uses the new credentials from a server, the movement will be detected if the legitimate user has not accessed that server recently. For example, in Figure 6.2, an attacker performs L_3 to access Server Y and manages to acquire Bob's credentials, e.g., because they remain cached in memory or in a local credential database, such as the SAM database on a Windows server. When the attacker then performs login L_4, the causality engine will infer a set of two-hop paths, where L_4 corresponds to the path's focal login, and the first hop of the path ranges over all logins into Y within the maximum session duration. If L_1 and L_2 both occurred sufficiently long ago (over the maximum session duration), then Hopper will infer a single causal path (L_3 to L_4) with causal user Alice and $\text{AllUsers}_P = \{\text{Alice}\}$. Since this inferred path involves a clear switch in credentials and accesses a destination Z that Alice has never previously accessed, Hopper generates an alert. If Alice has previously accessed Z, then Hopper adds this two-hop path to the watchlist: any future hops made along this path will create an extended path, which allows Hopper to reclassify the path and generate an alert once the attacker accesses a new destination.

Attack Scenario 2: Paths with Unclear Causality

The second attack scenario handles cases of "unclear causality", where Hopper infers multiple causal paths for a login, and some contain a credential switch and others do not. This happens when Hopper's path inference algorithm cannot tell which logins are causally linked, and thus which user is the login's true causal user. Revisiting the example in Figure 6.2, suppose both Bob's benign login (L_2) and the attack path's first hop (L_3) shortly before L_4. Then the causality engine would infer two causal paths for L_4: the attack path (L_3 to L_4) as well as a false benign path (L_2 to L_4).

More precisely, the second scenario covers the case where there are multiple causal paths for a login with target username U, where at least one path has a causal user that is not U, and at

least one path has U as its causal user and as the target username in every login. To handle these ambiguous situations, Hopper uses a probabilistic detection algorithm that quantifies the anomalousness (rareness) of each inferred path and alerts if any path is highly anomalous.

Alert Overview: Unclear Causality: Given an inferred path (P) with unclear causality, Hopper first checks whether the path ever visits a machine that its causal user (Alice) has not previously accessed in the historical training data (the second attack property). If Alice has access to all of the path's destinations, then Hopper adds this path to its causality engine's watchlist. Otherwise, Hopper runs the following anomaly detection algorithm on P.

First, Hopper extracts a set of three features that characterize the path's rareness. Next, Hopper uses P's features to compute a "suspiciousness" score for the path, which it then uses to rank P relative to a historical batch of paths (e.g., the past 30 days). If P ranks among the top $30\times B$ most suspicious historical paths, then Hopper generates an alert; otherwise, it adds the path to Hopper's watchlist. B corresponds to a user-provided budget that specifies the average number of daily alerts that an analyst has time to process for paths that might reflect this attack scenario. By looking for paths that are not only anomalous, but also contain key attack properties (a potential switch in credentials and new destination access), Hopper can significantly reduce the volume of alerts generated by traditional anomaly detection approaches.

Path Features: Hopper uses a set of historical "training" logins (e.g., the past 30 days) to extract the following three features for a path P. Let A refer to the path's starting machine, Z refer to the path's final destination, L_i refer to a login in P that switched from the causal user's credentials to a new set of credentials, and L_{i-1} refer to a login immediately preceding the credential-switching login in the path.

First, Hopper computes the historical edge frequency of L_{i-1}, where the frequency equals the number of *days* where a successful login with the exact same (source, destination, and target username) has occurred in the training data. Second, Hopper computes the historical edge frequency for each *subsequent* hop in the remainder of P and takes the lowest frequency value among these hops; i.e., the historical frequency of the rarest login starting at L_i until the path's final hop. Finally, Hopper computes the number of historical days where any successful login path (direct one-hop logins or any inferred multi-hop path) connects Machine A and Machine Z.

Intuitively, these three features estimate the rareness of a path in a length-agnostic manner, allowing Hopper to track and compare arbitrary length attack paths. The first feature helps Hopper identify when an attacker moves to a machine that users with additional privileges also access: users typically access a scoped set of servers related to their job function, and each internal server typically provides a specific service (i.e., team functionality). As a result, logins that enable an attacker to access credentials from other roles and job functions might rarely occur. The second feature captures the intuition that users typically have common workflows, and corresponding login paths, for accessing servers relevant to their job; so, it is unusual for two different team's logins to originate from the same intermediate server. In particular, we expect that login paths to sensitive servers will typically occur from a small set of source machines (e.g., either client machines or restricted servers that only similarly privileged users access); this behavior will cause attack paths from other users to have a suspicious (low) value for the second feature. The final feature captures

Algorithm 2 Hopper's anomaly scoring algorithm

Sub-Score(P, F, L):
1: $\text{Sum}_F \leftarrow 0$
2: $N \leftarrow 0$ (the total # of true causal paths)
3: **for** each path X in L **do:**
4: **if** P has a smaller value for F than X:
5: $\text{Sum}_F \leftarrow \text{Sum}_F + C_x$
where C_x = the path likelihood for X
6: $N \leftarrow N + C_x$,
7: $\text{Sub-Score}_F \leftarrow \text{Sum}_F \,/\, N$

Score(P, L): $\prod_F$ Sub-Score(P, F, L)

AlertGen(P, A (historical alerts), L (historical paths)):
1: **for** each path X in A **do:**
2: **if** Score(P, L) $\geq$ Score(X, L):
3: Alert on P

the overall rareness of a path in a length agnostic way: a path is more suspicious if its endpoint machines rarely have a path connecting them in prior history.

Anomaly Scoring: Given a path P and its features, Algorithm 2 shows the anomaly scoring procedure that Hopper uses to quantify the path's suspiciousness and make an alerting decision. Intuitively, Hopper's anomaly scoring algorithm generates an alert for P if it has one of the most suspicious feature sets in recent history.

Hopper's alerting algorithm (AlertGen) takes three inputs: a path to score (P), a set of historical paths (L) to compute P's anomaly score, and a set of historical alerts (A) for paths with unclear causality. To form a set of historical paths for P's anomaly score, Hopper iterates over each login in the historical training data and uses Hopper's causality engine to produce a set of all the two-hop paths for each historical login; the aggregate collection of all of these two-hop paths forms the historical path set (L). For efficiency, Hopper can compute this as a batch job at the beginning of each week, and reuse this set of historical paths for the entire week's scoring. The historical set of alert paths consists of the $B \times H$ most suspicious paths during the historical window, where H is the number of days in the historical window.

Given these three inputs, Hopper computes an overall anomaly score for P that represents the fraction of historical paths where P had more (or equally) suspicious feature values. Hopper then compares P's anomaly score against the scores for all historical alert paths, and generates an alert for P if its score exceeds any alert path's score; i.e., if P is at least as suspicious as a previous alert's path, Hopper produces an alert. If Hopper's scoring algorithm does not generate an alert for P, it adds the path to its watchlist for additional monitoring and potential reclassification.

Computing Scores: Conceptually, a path P's anomaly score corresponds to a cumulative tail probability: how much more suspicious (unlikely) is P relative to the kinds of paths that benign users historically make? As illustrated in the *Score* subroutine in Algorithm 2, Hopper computes this score by computing a sub-score for each of the path's features, and then multiplies these sub-scores to get an overall score. Each feature's sub-score computes the fraction of historical paths where P had a more suspicious feature value.

However, computing this sub-score is complicated by limitations in Hopper's path inference: the historical paths (L) that Hopper generates contain many falsely-inferred paths that do not correspond to a true causal path. For example, in Figure 6.2, if all four logins occur within 24 hours of each other, Hopper will generate a historical path set that contains three causal paths for L_4 — but only one of these paths reflects a true causal path. If we naively computed P's sub-score using the entire set of paths inferred by Hopper, then these scores will be skewed by logins that happen to occur from frequently-used servers; logins launched from these servers will naturally have many inbound hops, leading to many inferred causal paths whose features receive an inflated weight.

To account for these types of distributional biases, introduced by the imprecision in Hopper's path inference, each sub-score computes a weighted fraction of the number of historical paths where P had a more suspicious feature value. Specifically, as described in Section 6.5, whenever Hopper generates a set of paths for a login L_i in the historical dataset, it annotates each path with a "Path Likelihood" (denoted as C) that equals 1 divided by the total number of causal paths that Hopper inferred for L_i. When Hopper computes a sub-score for P, it uses C to down-weight the impact of each historical path on P's sub-score. In a naive computation without this weighting, Hopper would increment P's sub-score by 1 for each historical path where P had a more suspicious feature value. Instead, Hopper actually increments P's sub-score by a weighted fraction $1 \times C$ for each historical path where P had a more suspicious feature.

Alert Clustering

Hopper clusters its alerts each day to avoid generating redundant alerts for the same path. Whenever Hopper generates an alert for a path, it adds the path to a list of the current day's alerts. Before alerting on a new path, Hopper checks whether the new path has a focal login that matches the focal login of any path on the day's alert list; for any alert path, the focal hop captures a login that might involve a credential switch from the causal user. If so, Hopper updates the existing alert with information from the new path (e.g., expanding the alert path's hop list). Conceptually, this clustering corresponds to generating only one alert per day for each login where an attacker could have used a newly acquired set of credentials.

6.7 Evaluation

We evaluated Hopper using our 15-month dataset from Dropbox, measuring its detection rate (fraction of attacks detected) and the total volume of alerts it generates. Our data does not contain any known lateral movement attacks, but it does contain one in-situ lateral movement attack conducted

Attack Scenario	Detected Attacks (Clear switching)	Detected Attacks (Unclear Causality)	Detection Rate
Exploratory Attack	30	7	*37 / 41
Exploratory Attack (Stealthy)	6	62	68 / 69
Aggressive Spread	34	4	*38 / 41
Aggressive Spread (Stealthy)	35	34	69 / 69
Targeted Attack	28	10	*38 / 40
Targeted Attack (Stealthy)	5	57	62 / 67
Red Team Attack	0	1	1 / 1
All Attacks	138	174	312 / 327

Table 6.3: Summary of Hopper's attack detection performance (Section 6.6). The second column reports the number of attacks detected by Hopper's detector for clear-credential switching. The third column reports the number of attacks detected by Hopper's detector for paths with unclear causality. The last column reports Hopper's overall detection rate: the total number of attacks identified by any of Hopper's detectors, divided by the total number of attacks for the scenario. For the three rows of non-stealthy attacks marked with asterisks, Hopper failed to detect these attacks due to inaccurate or missing login attributes in our attack data; however, if these logins had contained the correct attribute information (as reported by up-to-date production data stores at Dropbox), Hopper would have detected all of these false negatives via its clear credential switching detector.

by Dropbox's professional red team. Additionally, we generated and injected a realistic set of 326 simulated attacks into our dataset for a more thorough evaluation; as described in Appendix C.3, these attacks span a diverse range of attacks and attacker stealthiness levels. Hopper can successfully detect 95.4% of the attacks in our dataset, including the red team attack, while generating an average of 9 alerts per day. On our dataset, we found that the best performing prior work [111] would need to produce over 8× as many alerts as Hopper to detect the same number of attacks.

Attack Data

Red Team Attack: Our dataset contains one lateral movement attack generated by a member of Dropbox's professional red team, which specializes in offensive security techniques and testing. The red team began their lateral movement from a "compromised" employee's laptop that they selected from a pre-existing pool of volunteer employees.[1] For their attack, they chose to simulate a common lateral movement scenario [43, 132], where an attacker conducts lateral movement to access to an organizations' domain controllers (credential and permission management

[1]The red team followed their standard safety protocols when conducting this simulation, which included recruiting and obtaining prior consent from all "compromised users", coordinating extensively with the security incident response team and various system administrators, and conducting any necessary remediation that resulted from the simulated attack (e.g., resetting any credentials that they accessed).

servers), providing them with widespread and persistent root privileges across the network. From their initial foothold, the red team conducted a series of reconnaissance and internal login (lateral movement) operations. Through this process, they identified and acquired a new, elevated set of credentials, which they then used to access one of the organization's domain controllers. Apart from the requirement that their movement needed to occur via credential-based logins (as opposed to exploiting a machine vulnerability), the red team performed this attack under no constraints or input from us. We reserved the red team data and did not examine it until we had frozen the design and parameters of our detector.

Realistic Attack Simulations: In addition to evaluating Hopper against the red team simulation, we developed an attack synthesis framework to generate an additional 326 realistic lateral movement attacks (Appendix C.3). At a high level, we randomly selected 50 employees in our data as starting victims, whose machines an attacker would compromise and use as an initial foothold to conduct lateral movement. For each of these starting victims, our framework synthesized twelve independent attack scenarios, corresponding to a pairing of one of three types of attacker goals with one of four levels of attacker stealthiness. The three attack goals simulate a ransomware attacker (an "aggressive spread" attack), an "exploratory attacker" who opportunistically uses a compromised machine to access additional internal machines, and a targeted attacker who aims to compromise a specific set of sensitive machines within the enterprise (e.g., an organization's domain controllers or critical infrastructure like DNS servers). The four levels of attacker stealthiness correspond to an attacker who only makes logins that traverse previously successful edges (i.e., does not make anomalous logins), an "active credential" attacker who only uses a set of credentials in a login if the legitimate user was recently logged into the source machine (i.e., resulting in login paths with ambiguous causality), an attacker who makes logins according to both of these stealthiness levels, and an attacker who does not deliberately make stealthy logins. The red team attack corresponded to a targeted attacker with "active credential" stealthiness, producing an attack path with unclear causality.

In total, we synthesized 326 attacks using this procedure, spanning the attack scenarios shown in Table C.1. As described in Appendix C.3, some simulations resulted in failed attacks that could not perform lateral movement under their constraints (e.g., because the randomly selected user did not have any stealthy paths to a sensitive machine). Nonetheless, this process produced a diverse set of 326 attacks that reflected many types of lateral movement described in a number of real-world APT reports, including stealthy paths that we describe later in this section.

Results

Evaluation Procedure: We divided our dataset into a training window (Jan 1, 2019 to Mar 1, 2019), which we used to bootstrap the feature extraction and scoring components of Hopper that require historical data, and a 13-month evaluation window (Mar 1, 2019 to Apr 1, 2020). Over the evaluation time window, the dataset contains 713,617,425 successful logins, and 2,941,173 logins after applying Hopper's data filtering steps (Section 6.3). As described below, we then used the logins in this evaluation window to compute Hopper's false positive rate and detection rate. For

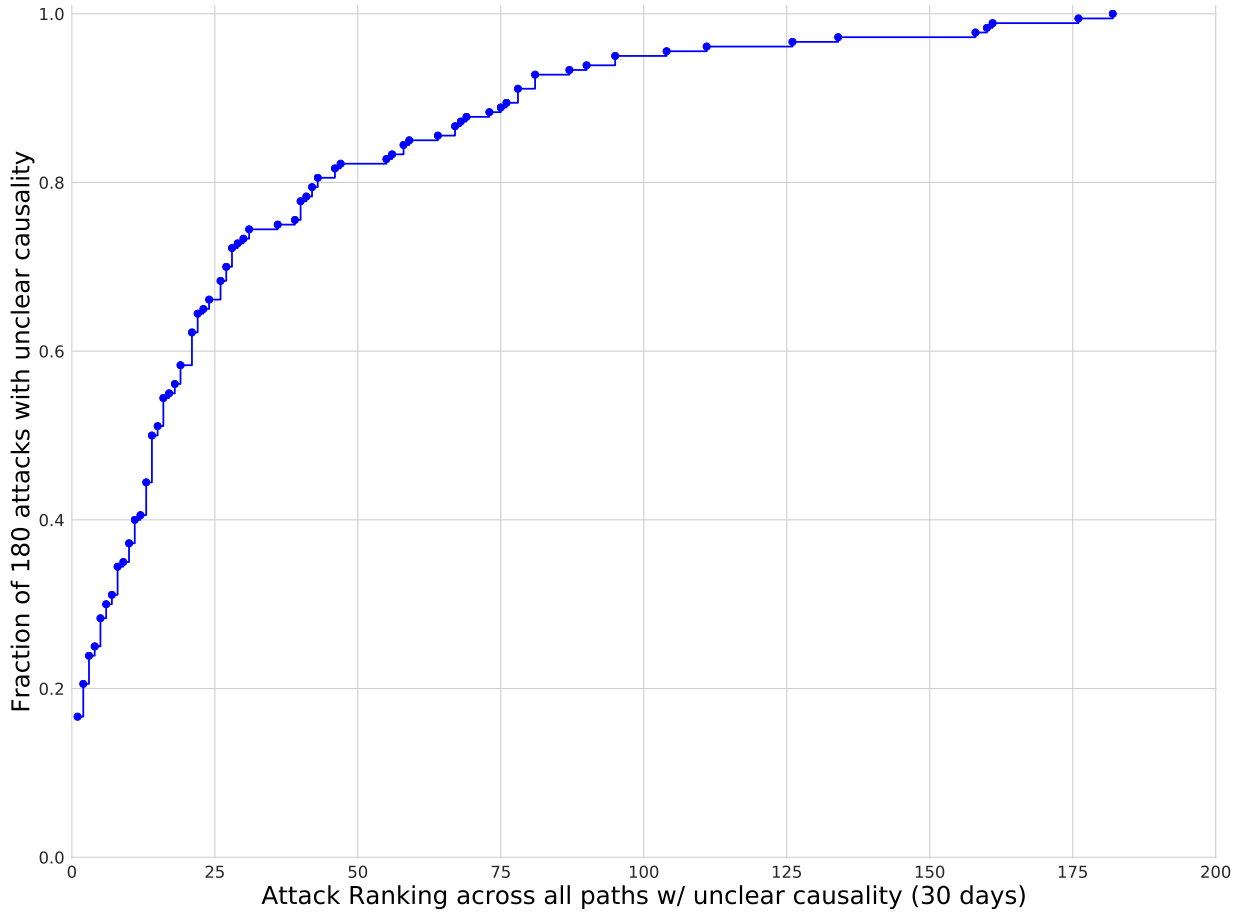

Figure 6.6: The ranking of attacks paths with unclear causality, relative to all alerts generated by Hopper during a 30-day window surrounding the attack. Hopper ranks over 75% of attack paths among the top 30 most-suspicious paths during the 30-day window.

any detection component that required historical training data, we used a rolling window of the preceding 30 days; for our anomaly scoring algorithm (Section 6.6), we used an initial budget of 5 alerts per day (we further explore the sensitivity of this parameter below).

Attack Detection Rate: For each of the 326 attacks our framework synthesized, we injected the attack's logins into our evaluation data and ran Hopper on the day(s) when the attack occurred. For the red team exercise, we examined the alerts that Hopper generated for the day of the attack. If Hopper generated an alert for any attack path made by the simulated attacker or red team, then we deemed Hopper successful at detecting the attack.

As shown in the last column of Table 6.3, Hopper successfully identified a total of 312 attacks, which includes the lateral movement attack performed by Dropbox's expert red team. Hopper detects 138 attacks through its detector for paths with clear-credential switching (Section 6.6). In all of these attacks (Table 6.3, Column 2), the simulated attacker either used a new set of credentials

in a login from their initial foothold machine or from a server that the legitimate user (of the new credentials) had not recently accessed, enabling Hopper to identify a movement path where the attacker switched to using an unexpected set of credentials. Because this component of Hopper's detector does not rely on the frequency of the individual graph edges created by attacker movement, it successfully detected several stealthy attacks where the adversary attempted to evade detection by only moving between machines with prior connection history.

However, the majority of our simulated attacks created paths with unclear causality, either because the attack quickly capitalized on new credentials that were recently used on a server, or because the attack simulated a stealthy adversary who attempted to avoid detection by only using new credentials from machines where the legitimate user was recently or currently active. Detecting these paths falls to Hopper's anomaly scoring detector (Section 6.6). Under our initial budget of 5 alerts per day, Hopper successfully identified 174 of these attacks, as well as the lateral movement attack simulated by Dropbox's red team.

False Negatives and Budget Sensitivity: Of the 15 attacks that Hopper failed to detect, we found that Hopper missed 9 attacks because of attribute errors in the login data we leveraged for this study. For each of these 9 false negatives, the attack logins that our framework synthesized had incorrect client versus server labels and/or missing or incorrect machine ownership information. When we replaced this incomplete or missing information in the attack logins with more accurate attribute labeling (acquired from up-to-date data sources at Dropbox), we found that Hopper could successfully detect all 9 of these false negatives with its clear credential switching detector.

Additionally, Hopper failed to detect 6 stealthy attacks under a daily budget of 5 alerts. If we increased the budget for Hopper's anomaly scoring detector by 1 additional alert per day, Hopper can successfully detect all-but-one of these false negatives; increasing the daily budget by 2 alerts per day would allow Hopper to detect all 6 false negatives. Further investigating the sensitivity of Hopper's detection rate to the user-provided alert budget, for all of the 180 synthesized attacks with uncertain causality, we computed the ranking of each attack's most suspicious path relative to all of the login paths in the 30 days surrounding each attack; i.e., the number of benign paths during the entire month that have a more suspicious score than an attack path. As shown in Figure 6.6, Hopper ranks over 75% of these attacks with unclear causality in the top 30 paths over the entire month in which the attack occurred and over 90% of attacks in the top 120 paths over the entire month (i.e., on average, the most suspicious alert and within the top four alerts per day respectively). This distribution indicates that Hopper's feature set and scoring algorithm consistently identify even these stealthy attacks with a low volume of false positives.

Total Alerts and False Positives: To compute Hopper's false positive rate, we ran Hopper on all legitimate (non-synthesized) logins for each of the 396 days in our evaluation data and aggregated the alerts it produced. Apart from the one red-team attack, Dropbox's incident database contains no instances of known lateral movement during our evaluation window. Thus, we conservatively labeled all of the alerts generated by Hopper on the login data as false positives, if they did not relate to the red team activity.

With a daily budget of 5 alerts for its unclear causality detector, Hopper generated 3,544 alerts across the 396-day evaluation window: an average of 9 alerts per day and an overall false positive

Detector	Detection Rate	Min. Total Alerts
SAL (CCS 2017 [111])	156 / 327	3,556
	312 / 327	28,771
Hopper	312 / 327	3,544

Table 6.4: Across our 13-month evaluation window (713M logins), Hopper generated 3,544 alerts to detect all known attacks. In contrast, SAL, the best performing prior work [111], required over 8× as many alerts to detect all of these attacks, and produced a similar number of alerts as Hopper to detect half as many attacks (Section 6.7).

rate of 0.000005 on the 713M raw logins in our evaluation data. Across the alerts, Hopper's detector for paths with a clear credential-switch produced 2,399 alerts, and the remaining 1,145 alerts resulted from Hopper's anomaly scoring detector. On some days, Hopper's anomaly scoring detector generated fewer than 5 alerts because (1) not every day had 5 sufficiently suspicious paths with causal uncertainty (e.g., weekends and holidays), and (2) our alert clustering resulted in some days with fewer alerts (Section 6.6).

Analyzing a random sample of Hopper's alerts, we identified a few general benign reasons that explain these likely-false positives. First, our anomaly scoring detector outputs a set of the most suspicious alerts each day; however, many of these paths reflect imprecision in Hopper's causal inference algorithm. In particular, we noticed that many days include alerts for paths that involve a potential credential switch between one system administrator (causal user) and another system administrator (the target user), which results from a few machines that these sysadmins frequently access and initiate logins from. Since these paths involve only administrator credentials, Hopper could reduce its false positives by filtering them out, since any credential switch between them likely provides limited additional access; future work could also explore using fine-grained host logging to resolve this causal ambiguity. Second, many of our alerts correspond to logins from client machines that (1) "switch" from the owner's credentials to using an infrequently-used service account or (2) reflect a system administrator running a re-provisioning script on an existing laptop to reassign it to a new user. Appendix C.4 describes these false positives in more detail.

Comparison with Prior State-of-the-Art

We compared Hopper's performance against the best performing prior work, the Structurally Anomalous Login (SAL) detector proposed by Siadati and Memon [111]. At a high level, SAL detects lateral movement attacks by generating a set of candidate alerts that consists of all logins that traverse a rare edge in the login graph (where rarity is a user-specified threshold). Next, SAL learns a set of "benign login patterns" that leverages additional properties about the machines and users involved in a login (e.g., the machine's type and/or the user's team). SAL then produces an alert for every candidate alert that does not match a benign login pattern. (Appendix C.5 provides a more detailed description of SAL.)

We applied SAL with a rolling two-month training window on all of the post-filtered logins in our evaluation window (i.e., the same data used for Hopper's evaluation and after applying both the data filtering and benign scenario pruning outlined in Section 6.3 and Section 6.6). As described in Appendix C.5, SAL takes two user-provided thresholds for training and classification, respectively. To compute the results summarized in Table 6.4, we explored a range of threshold combinations for SAL and selected the threshold combinations that produced the minimum volume of alerts to detect (1) all of the attacks in our data and (2) half of the attacks in our data. We report the number of alerts SAL produces after de-duplicating the alerts to only include one edge (source, destination, and target user) per day, and we considered SAL successful if it produced an alert for any of the malicious edges in an attack path. Appendix C.5 describes this procedure in more detail and shows SAL's performance under each combination we tried.

The large difference in alert volumes produced by SAL and Hopper stem from the fundamental differences in how they define lateral movement paths. SAL follows a traditional anomaly detection approach, which in theory, can detect any lateral movement attack given loose enough thresholds. Although this more general approach might detect attacks that Hopper cannot, our results suggest that Hopper can detect the prevalent class of lateral movement in our threat model with over 8$\times$ fewer false positives. Moreover, Hopper can successfully detect these attacks without the need for hand-tuned thresholds that SAL required for its optimal performance.

Attack Case Studies

Below, we describe and explore two of the attacks synthesized by our framework to illustrate the overlapping and different detection capabilities of Hopper and traditional anomaly detection approaches such as SAL. As described in Appendix C.3, our framework generates attacks based on the login graph of Dropbox by synthesizing lateral movement login data that begins from randomly selected employee machines (modeling common APT attacks that gain entry via a spear-phishing or water-holing attack).

Example Attack 1: Targeted Compromise: One attack simulated an adversary who began their lateral movement from an engineer's laptop and then attempted to access one of several high-value machines within an organization (e.g., a credential and permission management server). After three logins, the attacker arrived on a machine where a system administrator, Bob, had an active ssh session with a forwarded SSH agent. Our framework then synthesized a login that simulated the attacker abusing this agent to launch a fourth and final login into an authentication server that manages user permissions and SSH keys.

The last two logins involved in this attack path rarely occur, enabling SAL to detect this attack with a low volume of false positives. Similarly, Hopper successfully detects this attack, even though it involves an attack path with unclear causality (since the sysadmin had an active ssh session that could have launched the final login into the ssh management server); the rareness of the attack path's edges led Hopper to rank it among the top 10 most suspicious paths that month.

Example Attack 2: Stealthy, Short Paths: Another attack simulated a stealthy attacker who only accesses machines via previously traversed graph edges (e.g., by mining the shell and remote ses-

sion history of their current machine to identify where they can successfully move next). Starting from a compromised user (Alice)'s machine, the attacker identified a server (Y) which Alice had previously accessed during the past few months (4 out of the past 60 days) and synthesized a login along this prior edge.

After making this move to Server Y, the attacker identified all users who logged into Server Y within the past week. Our framework then simulated a scenario where the attacker observed that Server Y still had the credentials of a sysadmin, Bob, cached in its authentication database from a login during the past week, enabling the attacker to acquire them. Furthermore, the attacker (our framework) observed that Bob had also previously logged into a powerful remote management machine from Server Y (on 3 out of the past 60 days). Accordingly, our framework synthesized a final, second attack login using Bob's credentials to access this management server.

Although seemingly short and simple, this simulated attack actually reflects a realistic path that a stealthy, sophisticated attacker would make; shorter paths provide fewer opportunities for detection and require less time from the attacker.

Hopper successfully identified this attack under its clear-credential-switching path detector: the second hop of the attack switched to a new target username, but over 24 hours elapsed since Bob logged into Server Y. Even if Bob had logged into Server Y more recently, we observed that Hopper would still have caught this attack under its unclear-causal path detector (which ranks the attack path among the top 20 most suspicious in the past month). In contrast, because this attack only traverses edges with prior history, the best performing thresholds for SAL still produce a minimum of 14,000 alerts across our 13-month evaluation data.

This scenario highlights an advantage of our specification-based approach to anomaly detection over traditional anomaly detection. The latter implicitly assumes that attacks will necessarily involve rare logins between machines that infrequently communicate with each other. However, by mining and leveraging prior login history on a machine, a stealthy attacker can violate this assumption. By searching for paths that are not simply anomalous, but also exhibit key attack characteristics (the "specification"), Hopper can effectively detect these attacks while generating orders-of-magnitude fewer alerts.

6.8 Chapter Summary

This chapter presented Hopper, a system that detects lateral movement attacks by using commonly collected enterprise logs to create a graphical model of user login activity. We developed an efficient algorithm for inferring paths of causally-related logins, enabling Hopper to identify when an attacker's movement uses a suspicious set of credentials and accesses unusual machines. Leveraging a new anomaly scoring algorithm that requires no labeled data, Hopper can detect 312 / 327 lateral movement attacks in our 15-month real-world dataset, while generating an average of 9 alerts per day. In contrast, to find the same number of attacks, the prior state of the art would need to generate over 8$\times$ as many alerts as Hopper. Taken together, the work presented in these past two chapters illustrates a practical, data-centric avenue for mitigating attacks, even after they have managed to gain entry into an enterprise's internal environment.

Chapter 7

Conclusion and Future Directions

Organizations routinely fall victim to a range of sophisticated attacks, resulting in billions of dollars in financial harm, the theft of sensitive data, and the disruption of critical infrastructure and services [35, 41, 78, 85, 101]. This dissertation paves a promising path forward through a new set of data-driven insights and methods that can mitigate these damaging threats.

Two key ideas enable the systems in this dissertation to achieve practical success. First, we showed how to apply the principle of decomposition to sophisticated enterprise attacks: enabling us to develop a set of complementary defenses that target key elements of modern-day threats. In particular, our work presents an effective approach to deconstructing enterprise attacks into two key phases: how attackers gain access to an enterprise network and how attackers spread within an organization's internal environment. Extending this further, we also developed new conceptual models that capture the fundamental actions that successful attacks need to perform in each phase. Through this decomposition, our detection systems more effectively target the key characteristics of prevalent attacks. Second, this dissertation highlighted and addressed limitations in the traditional data-driven toolbox for security problems. Unlike many benign data settings, targeted enterprise attacks present unique challenges, such as the base rate fallacy, that hamper many traditional detection approaches: the combination of an attack's rare, stealthy nature and the deluge of diverse benign activity that naturally occurs in enterprises renders common learning-based approaches ineffective or impractical to use. The incredibly low base rate of these attacks means that a naive detector, which labels all events as benign, can achieve incredibly high overall accuracy in our problem settings, despite missing every attack. On the other hand, although standard machine learning techniques can often detect attacks with seemingly low error rates, the natural occurrence of the attacks we studied is often orders of magnitude smaller than the false positive rates obtained by these canonical methods. As a result, these traditional approaches produce an overwhelming and intractable volume of false positives. To overcome these base rate challenges, we showed how to redesign anomaly detection techniques, so that they leverage security domain knowledge to search for events that are not simply anomalous, but explicitly suspicious. Because of their more refined search criteria, the techniques that result from our new "specification-based anomaly detection" approach can achieve orders-of-magnitude fewer false positives, enabling practical detection.

In sum, this dissertation builds a promising foundation for data-driven defenses against the constantly evolving landscape of enterprise attacks. The collaborations we undertook with real-world organizations throughout this dissertation enabled us to develop empirically-grounded insights and assess the practicality of our methods. Coupled with the advances enabled by our two key ideas, this grounded realism has already led to practical adoption of our work: government-funded organizations such as the Lawrence Berkeley National Laboratory, commercial security vendors such as Barracuda Networks, and large companies such as Facebook and Dropbox have all implemented and used many of the ideas we developed to uncover real-world attacks and improve their security.

As parting thoughts, we conclude this dissertation by sketching three promising directions for future work.

New Directions in Anomaly Detection for Security: This dissertation highlighted a key gap in our existing technical toolbox: effective methods for uncovering rare attacks in a dataset replete with anomalous-but-benign noise. Although we have made progress on this front through our new approach to anomaly detection, a number of exciting directions remain. From a practical perspective, many organizations deploy a variety of detectors that use manually-tuned thresholds, and, occasionally, some form of traditional anomaly detection to identify attacks. We hypothesize that many of these settings might benefit from developing or applying some of the specification-based anomaly detection methods we develop, such as DAS. New work could also explore ways to improve or build entirely new specification-based anomaly detection algorithms. In particular, the detection techniques we presented in Chapters 3 and 6 leverage a common characteristic in the way that security practitioners have traditionally designed signals and features for detection: many features used for detecting attacks are monotonically suspicious, e.g., where smaller values indicate more suspicious events and larger values indicate likely benign events, or vice-versa. By incorporating notions of monotonicity into their anomaly scores, the algorithms we developed significantly outperformed traditional detection methods because our methods focus only on particularly interesting regions of the feature space. Future work can explore whether there exist other thematic characteristics about the features or rule sets that existing detection strategies employ, but which traditional learning methods do not easily incorporate. On a more theoretical front, future work could study different statistical approaches to model an event's suspiciousness, such as more complex methods to incorporate or learn probabilistic distributions of how anomalous and malicious an event is. The algorithms we introduced, such as DAS, use simple metrics of density and cumulative frequency, which suffice for the problems we study; however, other security settings might require more sophisticated models, and such models might also lead to improvements in the systems we developed.

Integrating End-Users into the Alerting Process: Although the systems we develop in this dissertation generate low volumes of false alarms, they still add to an ever-growing list of alerts and incidents that security teams need to analyze and potentially remediate. To alleviate this workload, future research could explore how a detection system might securely delegate the job of triaging a potential incident to the end-user whose account or device triggered an alert. In particular, we think detecting enterprise social engineering attacks presents a promising space for such systems,

because end-users often have insightful context about what triggered an alert that might not be easily discernible by a detector or security analyst.

For example, in social engineering attacks like spearphishing, adversaries exploit a gap between an end-user's perception and the true identity of the correspondent with whom the user interacts. While a detector often has sufficient data to compute the true identity of an email's sender, it can be difficult to infer what entity the end-user *believes* they are interacting with and whether their perception of the sender's identity matches the actual identity that a detector has computed.

A human-in-the-loop detector could resolve this semantic gap by asking the end-user a minimal set of non-technical, fact-based questions to determine with whom the employee believes they are interacting, and then compute whether the user's perception presents a heightened risk in the context of the sender and email they have received. By reserving the technical and security decision making logic for a detection algorithm, and only relying on end-users to provide hard-to-infer, but non-technical observations, such an interactive detection system could make more accurate decisions.

Full Network Causality: The lateral movement work we presented in Chapter 6 highlights the power of tracing and understanding the flows of user movement between machines within an organization's internal environment. Extending this work to understand whether any internal movement paths have suspicious, and potentially causal, interactions with external entities presents an exciting direction for future work. In particular, in order to direct a compromised internal machine to conduct lateral movement within an enterprise, attackers will need to establish a remote connection with the machines they have compromised in order to "command and control" them. Thus, organizations could better detect and thwart attacker lateral movement if they can successfully identify when an enterprise machine suddenly engages in suspicious internal movement that appears to be directed and caused by an external entity. Similarly, in attacks that aim to steal data from an organization, attackers will need to establish some remote connection from an internal machine to an external party in order to exfiltrate any stolen data. Developing new methods to accurately identify machines that exhibit signs of data exfiltration, and correlating such network activity with signals about internal lateral movement, also presents an exciting avenue for uncovering and thwarting enterprise attacks.

Ultimately, this dissertation demonstrates the power of a methodical data-driven approach to security, and the value of re-imagining and redesigning existing techniques to fit the problem at hand. Despite the enormous challenges of detecting sophisticated attacks in messy, complex environments, our work illustrates that organizations can uncover and thwart these threats in a practical manner.

Bibliography

[1] S. Abu-Nimeh, D. Nappa, X. Wang, and S. Nair. A comparison of machine learning techniques for phishing detection. In Proceedings of the anti-phishing working groups 2nd annual eCrime researchers summit, pages 60–69. ACM, 2007.

[2] R. Allbery. remctl: Remote authenticated command execution. `https://github.com/rra/remctl`, 2018.

[3] K. Allix, T. F. Bissyandé, J. Klein, and Y. Le Traon. Are Your Training Datasets Yet Relevant? In Proc. of 7th Springer ESSoS, 2015.

[4] S. Axelsson. The base-rate fallacy and the difficulty of intrusion detection. ACM Transactions on Information and System Security (TISSEC), 2000.

[5] A. Bergholz, J. H. Chang, G. Paaß, F. Reichartz, and S. Strobel. Improved phishing detection using model-based features. In CEAS, 2008.

[6] J. Bergstra and Y. Bengio. Random search for hyper-parameter optimization. JMLR, 13(Feb), 2012.

[7] S. Bird, E. Loper, and E. Klein. Natural Language Toolkit. `https://www.nltk.org/`, 2019.

[8] A. Bohara, M. A. Noureddine, A. Fawaz, and W. H. Sanders. An unsupervised multi-detector approach for identifying malicious lateral movement. In IEEE 36th Symposium on Reliable Distributed Systems (SRDS), 2017.

[9] X. Bouwman, H. Griffioen, J. Egbers, C. Doerr, B. Klievink, and M. van Eeten. A different cup of TI? the added value of commercial threat intelligence. In USENIX Security 20, 2020.

[10] A. Breeden, S. Chan, and N. Perlroth. Macron campaign says it was target of 'massive' hacking attack. `https://www.nytimes.com/2017/05/05/world/europe/france-macron-hacking.html`, May 2017.

[11] P. Bright. Spearphishing + zero-day: RSA hack not "extremely sophisticated". `http://arstechnica.com/security/2011/04/spearphishing-0-day-rsa-hack-not-extremely-sophisticated/`, April 2011.

[12] E. Bursztein, B. Benko, D. Margolis, T. Pietraszek, A. Archer, A. Aquino, A. Pitsillidis, and S. Savage. Handcrafted Fraud and Extortion: Manual Account Hijacking in the Wild. In Proc. of 14th ACM IMC, 2014.

[13] E. Bursztein and V. Eranti. Internet-wide efforts to fight email phishing are working. https://security.googleblog.com/2013/12/internet-wide-efforts-to-fight-email.html, Feb 2016.

[14] CarbonBlack. Global threat report: Year of the next-gen cyberattack. https://www.carbonblack.com/resources/threat-research/year-of-the-next-gen-cyberattack/, 2019.

[15] CERT. Advanced persistent threat activity targeting energy and other critical infrastructure sectors. https://www.us-cert.gov/ncas/alerts/TA17-293A, 2017.

[16] CERT. Ransomware. https://www.us-cert.gov/Ransomware, 2019.

[17] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. ACM Comput. Surv., 41(3):15:1–15:58, 2009.

[18] N. Chawla, N. Japkowicz, and A. Kotcz. Editorial: special issue on learning from imbalanced data sets. ACM SIGKDD Explorations Newsletter, 6(1):1–6, 2004.

[19] J. Chen, V. Paxson, and J. Jiang. Composition kills: A case study of email sender authentication. In USENIX Security Symposium (USENIX), 2020.

[20] A. Cidon. Threat Spotlight: Office 365 Account Takeover — the New "Insider Threat". https://blog.barracuda.com/2017/08/30/threat-spotlight-office-365-account-compromise-the-new-insider-threat/, Aug 2017.

[21] A. Cidon, L. Gavish, I. Bleier, N. Korshun, M. Schweighauser, and A. Tsitkin. High precision detection of business email compromise. In USENIX Security Symposium (USENIX), 2019.

[22] A. Cidon, L. Gavish, I. Bleier, N. Korshun, M. Schweighauser, and A. Tsitkin. High Precision Detection of Business Email Compromise. In Proc. of 28th Usenix Security, 2019.

[23] Cisco. What is network segmentation? https://www.cisco.com/c/en/us/products/security/what-is-network-segmentation.html, 2019.

[24] CloudMark. Spear phishing: The top ten worst cyber attacks. https://blog.cloudmark.com/wp-content/uploads/2016/01/cloudmark_top_ten_infographic.png.

[25] D. Community. Discontinuing rendering of html content. https://www.dropboxforum.com/t5/Manage-account/Discontinuing-rendering-of-HTML-content/td-p/187920, Sep 2016.

[26] L. Constantin. Phishing attacks that bypass 2-factor authentication are now easier to execute. https://www.csoonline.com/article/3399858/phishing-attacks-that-bypass-2-factor-authentication-are-now-easier-to-execute.html, June 2019.

[27] CrowdStrike. Lateral movement. https://www.crowdstrike.com/epp-101/lateral-movement/, Sep 2019.

[28] A. Dahan. Operation cobalt kitty, 2017.

[29] S. Duman, K. Kalkan-Cakmakci, M. Egele, W. Robertson, and E. Kirda. Emailprofiler: Spearphishing filtering with header and stylometric features of emails. In Computer Software and Applications Conference (COMPSAC), 2016 IEEE 40th Annual, pages 408–416. IEEE, 2016.

[30] J. Dunagan, A. X. Zheng, and D. R. Simon. Heat-ray: combating identity snowball attacks using machinelearning, combinatorial optimization and attack graphs. In ACM SIGOPS 22nd Symposium on Operating Systems Principles (SOSP), 2009.

[31] J. Dutson, D. Allen, D. Eggett, and K. Seamons. Don't punish all of us: Measuring user attitudes about two-factor authentication. In IEEE European Symposium on Security and Privacy Workshops (EuroS&PW), 2019.

[32] M. Egele, G. Stringhini, C. Kruegel, and G. Vigna. COMPA: Detecting Compromised Accounts on Social Networks. In Proc. of 20th ISOC NDSS, 2013.

[33] FBI. BUSINESS E-MAIL COMPROMISE THE 12 BILLION DOLLAR SCAM, Jul 2018. https://www.ic3.gov/media/2018/180712.aspx.

[34] I. Fette, N. Sadeh, and A. Tomasic. Learning to detect phishing emails. In Proceedings of the 16th international conference on World Wide Web, pages 649–656. ACM, 2007.

[35] J. Finkle and S. Heavey. Target says it declined to act on early alert of cyber breach. http://www.reuters.com/article/us-target-breach-idUSBREA2C14F20140313, Mar 2014.

[36] S. Freitas, A. Wicker, D. H. Chau, and J. Neil. D2m: Dynamic defense and modeling of adversarial movement in networks. In Proceedings of the 2020 SIAM International Conference on Data Mining, 2020.

[37] S. Garera, N. Provos, M. Chew, and A. D. Rubin. A framework for detection and measurement of phishing attacks. In Proceedings of the 2007 ACM workshop on Recurring malcode, 2007.

[38] H. Gascon, S. Ullrich, B. Stritter, and K. Rieck. Reading between the lines: content-agnostic detection of spear-phishing emails. In International Symposium on Research in Attacks, Intrusions, and Defenses (RAID), 2018.

[39] S. Gatlan. Microsoft shares tactics used in human-operated ransomware attacks. https://www.bleepingcomputer.com/news/security/microsoft-shares-tactics-used-in-human-operated-ransomware-attacks/, Mar 2020.

[40] Google. Classification: ROC and AUC. https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc, 2019.

[41] R. Hackett. Anthem, a major health insurer, suffered a massive hack. http://fortune.com/2015/02/05/anthem-suffers-hack/, February 2015.

[42] A. Hagberg, N. Lemons, A. Kent, and J. Neil. Connected Components and Credential Hopping in Authentication Graphs. In 2014 Tenth International Conference on Signal-Image Technology and Internet-Based Systems, 2014.

[43] A. Hanel. Big game hunting with ryuk: Another lucrative targeted ransomware. https://www.crowdstrike.com/blog/big-game-hunting-with-ryuk-another-lucrative-targeted-ransomware/, Jan 2019.

[44] W. U. Hassan, A. Bates, and D. Marino. Tactical provenance analysis for endpoint detection and response systems. In IEEE Symposium on Security & Privacy 20, 2020.

[45] W. U. Hassan, M. A. Noureddine, P. Datta, and A. Bates. Omegalog: High-fidelity attack investigation via transparent multi-layer log analysis. In Network and Distributed System Security Symposium, 2020.

[46] S. Hawley, B. Read, C. Brafman-Kittner, N. Fraser, A. Thompson, Y. Rozhansky, and S. Yashar. Apt39: An iranian cyber espionage group focused on personal information. https://www.fireeye.com/blog/threat-research/2019/01/apt39-iranian-cyber-espionage-group-focused-on-personal-information.html, Jan 2019.

[47] H. He and E. A. Garcia. Learning from imbalanced data. IEEE Transactions on knowledge and data engineering, 21(9):1263–1284, 2009.

[48] G. Ho, A. Cidon, L. Gavish, M. Schweighauser, V. Paxson, S. Savage, G. M. Voelker, and D. Wagner. Detecting and characterizing lateral phishing at scale. In Proc. of 28th Usenix Security, 2019.

[49] G. Ho, A. Sharma, M. Javed, V. Paxson, and D. Wagner. Detecting credential spearphishing in enterprise settings. In USENIX Security 17, pages 469–485, 2017.

[50] M. N. Hossain, S. Sheikhi, and R. Sekar. Combating dependence explosion in forensic analysis using alternative tag propagation semantics. In IEEE Symposium on Security & Privacy 20, 2020.

[51] M. N. Hossain, J. Wang, O. Weisse, R. Sekar, D. Genkin, B. He, S. D. Stoller, G. Fang, F. Piessens, and E. Downing. Dependence-preserving data compaction for scalable forensic analysis. In USENIX Security 18, 2018.

[52] H. Hu and G. Wang. End-to-end measurements of email spoofing attacks. In USENIX Security Symposium (USENIX), 2018.

[53] X. Hu, B. Li, Y. Zhang, C. Zhou, and H. Ma. Detecting Compromised Email Accounts from the Perspective of Graph Topology. In Proc. of 11th ACM CFI, 2016.

[54] D. Hubbard. Cisco Umbrella 1 Million. `https://umbrella.cisco.com/blog/2016/12/14/cisco-umbrella-1-million/`, Dec 2016.

[55] T. N. Jagatic, N. A. Johnson, M. Jakobsson, and F. Menczer. Social phishing. Communications of the ACM, 2007.

[56] N. Johnston. Dropbox users targeted by phishing scam hosted on dropbox. `https://www.symantec.com/connect/blogs/dropbox-users-targeted-phishing-scam-hosted-dropbox`, Oct 2014.

[57] C. Kanich, C. Kreibich, K. Levchenko, B. Enright, G. M. Voelker, V. Paxson, and S. Savage. Spamalytics: An Empirical Analysis of Spam Marketing Conversion. In Proc. of 15th ACM CCS, 2008.

[58] T. Karagiannis and M. Vojnovic. Email information flow in large-scale enterprises. Technical report, Microsoft Research, 2008.

[59] A. D. Kent, L. M. Liebrock, and J. C. Neil. Authentication graphs: Analyzing user behavior within an enterprise network. Computers & Security, 2015.

[60] M. Khonji, Y. Iraqi, and A. Jones. Mitigation of spear phishing attacks: A content-based authorship identification framework. In Internet Technology and Secured Transactions (ICITST), 2011 International Conference for, pages 416–421. IEEE, 2011.

[61] P. Kumaraguru, J. Cranshaw, A. Acquisti, L. Cranor, J. Hong, M. A. Blair, and T. Pham. School of phish: a real-world evaluation of anti-phishing training. In Proceedings of the 5th Symposium on Usable Privacy and Security, page 3. ACM, 2009.

[62] F. Labs. A sobering day. `https://labs.ft.com/2013/05/a-sobering-day/?mhq5j=e6`, May 2013.

[63] A. Lazarevic, L. Ertoz, V. Kumar, A. Ozgur, and J. Srivastava. A comparative study of anomaly detection schemes in network intrusion detection. In Proceedings of the 2003 SIAM International Conference on Data Mining, pages 25–36. SIAM, 2003.

[64] S. Le Blond, C. Gilbert, U. Upadhyay, M. G. Rodriguez, and D. Choffnes. A broad view of the ecosystem of socially engineered exploit documents. In NDSS, 2017.

[65] S. Le Blond, A. Uritesc, C. Gilbert, Z. L. Chua, P. Saxena, and E. Kirda. A look at targeted attacks through the lense of an ngo. In USENIX Security, pages 543–558, 2014.

[66] R. Lee, M. Assante, and T. Conway. Analysis of the cyber attack on the ukrainian power grid. Electricity Information Sharing and Analysis Center (E-ISAC), 2016.

[67] R. M. Lee, M. J. Assante, and T. Conway. Analysis of the cyber attack on the ukrainian power grid. https://ics.sans.org/media/E-ISAC_SANS_Ukraine_DUC_5.pdf, Mar 2016.

[68] F. Liu, Y. Wen, D. Zhang, X. Jiang, X. Xing, and D. Meng. Log2vec: A Heterogeneous Graph Embedding Based Approach for Detecting Cyber Threats within Enterprise. In Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, 2019.

[69] Q. Liu, J. W. Stokes, R. Mead, T. Burrell, I. Hellen, J. Lambert, A. Marochko, and W. Cui. Latte: Large-scale lateral movement detection. In IEEE Military Communications Conference (MILCOM), 2018.

[70] Mailgun Team. Talon. https://github.com/mailgun/talon, 2018.

[71] Mandiant. Apt1: Exposing one of china's cyber espionage units. https://www.fireeye.com/content/dam/fireeye-www/services/pdfs/mandiant-apt1-report.pdf, 2013.

[72] W. R. Marczak, J. Scott-Railton, M. Marquis-Boire, and V. Paxson. When governments hack opponents: A look at actors and technology. In USENIX Security, pages 511–525, 2014.

[73] Microsoft Graph: message resource type. https://developer.microsoft.com/en-us/graph/docs/api-reference/v1.0/resources/message. Accessed: 2018-11-01.

[74] Microsoft. People overview - Outlook Web App. https://support.office.com/en-us/article/people-overview-outlook-web-app-5fe173cf-e620-4f62-9bf6-da5041f651bf. Accessed: 2018-11-01.

[75] S. M. Milajerdi, R. Gjomemo, B. Eshete, R. Sekar, and V. Venkatakrishnan. Holmes: real-time apt detection through correlation of suspicious information flows. In IEEE Symposium on Security & Privacy, 2019.

[76] B. Miller, A. Kantchelian, M. C. Tschantz, S. Afroz, R. Bachwani, R. Faizullabhoy, L. Huang, V. Shankar, T. Wu, G. Yiu, et al. Reviewer Integration and Performance Measurement for Malware Detection. In Proc. of 13th Springer DIMVA, 2016.

[77] MITRE. Mitre att&ck mat. https://attack.mitre.org/, 2015–2019.

[78] E. Nakashima. Chinese breach data of 4 million federal workers. https://www.washingtonpost.com/world/national-security/chinese-hackers-breach-federal-governments-personnel-office/2015/06/04/889c0e52-0af7-11e5-95fd-d580f1c5d44e_story.html, June 2015.

[79] NCSC. Joint report on publicly available hacking tools. https://www.ncsc.gov.uk/report/joint-report-on-publicly-available-hacking-tools, 2018.

[80] A. Niakanlahiji, J. Wei, M. R. Alam, Q. Wang, and B.-T. Chu. Shadowmove: A stealthy lateral movement strategy. In USENIX Security 20, 2020.

[81] S. Nichols. It has been 15 years, and we're still reporting homograph attacks. https://www.theregister.com/2020/03/04/homograph_attacks_still_happening/, Mar 2020.

[82] Novetta. Operation SMN: Axiom Threat Actor Group Report. http://www.novetta.com/wp-content/uploads/2014/11/Executive_Summary-Final_1.pdf, Nov 2014.

[83] A. Oest, Y. Safaei, P. Zhang, B. Wardman, K. Tyers, Y. Shoshitaishvili, A. Doupé, and G.-J. Ahn. PhishTime: Continuous Longitudinal Measurement of the Effectiveness of Anti-phishing Blacklists. In USENIX Security Symposium (USENIX), 2020.

[84] A. Oest, P. Zhang, B. Wardman, E. Nunes, J. Burgis, A. Zand, K. Thomas, A. Doupé, and G.-J. Ahn. Sunrise to sunset: Analyzing the end-to-end life cycle and effectiveness of phishing attacks at scale. In USENIX Security Symposium (USENIX), 2020.

[85] T. C. of Economic Advisors. The cost of malicious cyber activity to the u.s. economy. https://www.whitehouse.gov/wp-content/uploads/2018/03/The-Cost-of-Malicious-Cyber-Activity-to-the-U.S.-Economy.pdf, Mar 2018.

[86] D. Oliveira, H. Rocha, H. Yang, D. Ellis, S. Dommaraju, M. Muradoglu, D. Weir, A. Soliman, T. Lin, and N. Ebner. Dissecting spear phishing emails for older vs young adults: On the interplay of weapons of influence and life domains in predicting susceptibility to phishing. In ACM Conference on Human Factors in Computing Systems (CHI), 2017.

[87] J. Onaolapo, E. Mariconti, and G. Stringhini. What Happens After You Are Pwnd: Understanding the Use of Leaked Webmail Credentials in the Wild. In Proc. of 16th ACM IMC, 2016.

[88] D. Organization. DMARC. https://dmarc.org/, 2016.

[89] J. Palme. Common Internet Message Headers. https://tools.ietf.org/html/rfc2076.

[90] F. Pendlebury, F. Pierazzi, R. Jordaney, J. Kinder, and L. Cavallaro. Tesseract: Eliminating experimental bias in malware classification across space and time. In Proc. of 28th Usenix Security, 2019.

[91] A. Peterson. The sony pictures hack, explained. https://www.washingtonpost.com/news/the-switch/wp/2014/12/18/the-sony-pictures-hack-explained/, Dec 2014.

[92] A. Picchi. Ransomware's mounting toll: Delayed surgeries and school closures. https://www.cbsnews.com/news/ransomware-attack-621-hospitals-cities-and-schools-hit-so-far-in-2019/, Oct 2019.

[93] F. Plan, N. Fraser, J. O'Leary, V. Cannon, and B. Read. Apt40: Examining a china-nexus espionage actor. https://www.fireeye.com/blog/threat-research/2019/03/apt40-examining-a-china-nexus-espionage-actor.html, Mar 2019.

[94] K. Poulsen. Google disrupts chinese spear-phishing attack on senior u.s. officials. https://www.wired.com/2011/06/gmail-hack/, Jul 2011.

[95] P. Prakash, M. Kumar, R. R. Kompella, and M. Gupta. Phishnet: predictive blacklisting to detect phishing attacks. In INFOCOM, 2010 Proceedings IEEE, pages 1–5. IEEE, 2010.

[96] A. Press. Ransomware attack cripples san bernardino city unified school district's computer system. https://abc7.com/ransomware-attack-cripples-san-bernardino-school-districts-computer-system/5635301/, Oct 2019.

[97] E. Purvine, J. R. Johnson, and C. Lo. A graph-based impact metric for mitigating lateral movement cyber attacks. In ACM Workshop on Automated Decision Making for Active Cyber Defense, 2016.

[98] S. Ragan. Office 365 phishing attacks create a sustained insider nightmare for it. https://www.csoonline.com/article/3225469/office-365-phishing-attacks-create-a-sustained-insider-nightmare-for-it.html, Sep 2017.

[99] J. S. Railton and K. Kleemola. London calling: Two-factor authentication phishing from iran. https://citizenlab.org/2015/08/iran_two_factor_phishing/, August 2015.

[100] F. Y. Rashid. Don't like Mondays? Neither do attackers. https://www.csoonline.com/article/3199997/don-t-like-mondays-neither-do-attackers.html, Aug 2017.

[101] S. Reilly. Records: Energy department struck by cyber attacks. http://www.usatoday.com/story/news/2015/09/09/cyber-attacks-doe-energy/71929786/, September 2015.

[102] Retraining models on new data. https://docs.aws.amazon.com/machine-learning/latest/dg/retraining-models-on-new-data.html, 2019.

[103] J. Reynolds, N. Samarin, J. Barnes, T. Judd, J. Mason, M. Bailey, and S. Egelman. Empirical measurement of systemic 2fa usability. In USENIX Security Symposium (USENIX), 2020.

[104] A. Robbin, R. Vazarkar, and W. Schroeder. Bloodhound: Six degrees of domain admin. https://bloodhound.readthedocs.io/en/latest/index.html/, 2020.

[105] J. J. Roberts. Homeland Security Chief Cites Phishing as Top Hacking Threat. http://fortune.com/2016/11/20/jeh-johnson-phishing/, Nov 2016.

[106] N. S. Safa, R. Von Solms, and S. Furnell. Information security policy compliance model in organizations. computers & security, 56, 2016.

[107] M. S. Schmidt and D. E. Sanger. Russian hackers read obama's unclassified emails, officials say. http://www.nytimes.com/2015/04/26/us/russian-hackers-read-obamas-unclassified-emails-officials-say.html, Apr 2015.

[108] S. Sheng, M. Holbrook, P. Kumaraguru, L. F. Cranor, and J. Downs. Who falls for phish?: a demographic analysis of phishing susceptibility and effectiveness of interventions. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pages 373–382. ACM, 2010.

[109] S. Sheng, B. Magnien, P. Kumaraguru, A. Acquisti, L. F. Cranor, J. Hong, and E. Nunge. Anti-phishing phil: the design and evaluation of a game that teaches people not to fall for phish. In Usenix Symposium on Usable Privacy and Security, 2007.

[110] S. Sheng, B. Wardman, G. Warner, L. Cranor, J. Hong, and C. Zhang. An empirical analysis of phishing blacklists. In Sixth Conference on Email and Anti-Spam (CEAS), 2009.

[111] H. Siadati and N. Memon. Detecting structurally anomalous logins within enterprise networks. In ACM SIGSAC Conference on Computer and Communications Security (CCS), 2017.

[112] A. Spark. PySpark DecisionTreeClassificationModel v2.1.0. http://spark.apache.org/docs/2.1.0/api/python/pyspark.ml.html.

[113] D. R. Staff. Fbi: Phishing can defeat two-factor authentication. https://www.darkreading.com/attacks-breaches/fbi-phishing-can-defeat-two-factor-authentication/d/d-id/1336070, Oct 2019.

[114] G. Stringhini and O. Thonnard. That ain't you: Blocking spearphishing through behavioral modelling. In Detection of Intrusions and Malware, and Vulnerability Assessment, pages 78–97. Springer, 2015.

[115] B. E. Strom, A. Applebaum, D. P. Miller, K. C. Nickels, A. G. Pennington, and C. B. Thomas. Mitre att&ck: Design and philosophy. Technical report, 2018.

[116] C. T. U. R. Team. Bronze union: Cyberespionage persists despite disclosures. https://www.secureworks.com/research/bronze-union, Jun 2017.

[117] K. Thomas, F. Li, C. Grier, and V. Paxson. Consequences of Connectivity: Characterizing Account Hijacking on Twitter. In Proc. of 21st ACM CCS, 2014.

[118] Trend Micro. Spear-phishing email: Most favored APT attack bait. http://www.trendmicro.com.au/cloud-content/us/pdfs/security-intelligence/white-papers/wp-spear-phishing-email-most-favored-apt-attack-bait.pdf, 2012.

[119] TrendMicro. Lateral movement: How do threat actors move deeper into your network? http://about-threats.trendmicro.com/cloud-content/us/ent-primers/pdf/tlp_lateral_movement.pdf, 2013.

[120] L. Tung. Ransomware: These sophisticated attacks are delivering 'devastating' payloads, warns microsoft. https://www.zdnet.com/article/ransomware-these-sophisticated-attacks-are-delivering-devastating-payloads-warns-microsoft/, Mar 2020.

[121] L. Vaas. How hackers broke into John Podesta, DNC Gmail accounts. https://nakedsecurity.sophos.com/2016/10/25/how-hackers-broke-into-john-podesta-dnc-gmail-accounts/, October 2016.

[122] N. Wetsman. Woman dies during a ransomware attack on a german hospital. https://www.theverge.com/2020/9/17/21443851/death-ransomware-attack-hospital-germany-cybersecurity, Sep 2020.

[123] C. Whittaker, B. Ryner, and M. Nazif. Large-scale automatic classification of phishing pages. In NDSS, volume 10, 2010.

[124] Wikipedia. Domainkeys identified mail. https://en.wikipedia.org/wiki/DomainKeys_Identified_Mail.

[125] Wikipedia. Server name indication. https://en.wikipedia.org/wiki/Server_Name_Indication, June 2017.

[126] Wikipedia. Network segmentation. https://en.wikipedia.org/wiki/Network_segmentation, Sep 2019.

[127] Wikipedia. Random forest. https://en.wikipedia.org/wiki/Random_forest, 2019.

[128] Wikipedia. Sender policy framework. https://en.wikipedia.org/wiki/Sender_Policy_Framework, Oct 2020.

[129] F. Wilkens, S. Haas, D. Kaaser, P. Kling, and M. Fischer. Towards Efficient Reconstruction of Attacker Lateral Movement. In Conference on Availability, Reliability and Security - ARES, 2019.

[130] D. Wind. Sophisticated Spear Phishing Campaigns using Homograph Attacks. https://www.offensity.com/de/blog/sophisticated-spear-phishing-campaigns-using-homograph-attacks/, May 2019.

[131] K. Zetter. Researchers uncover rsa phishing attack, hiding in plain sight. https://www.wired.com/2011/08/how-rsa-got-hacked/, Aug 2011.

[132] K. Zetter. Inside the cunning, unprecedented hack of ukraine's power grid. https://www.wired.com/2016/03/inside-cunning-unprecedented-hack-ukraines-power-grid/, Mar 2016.

[133] P. Zhang, A. Oest, H. Cho, R. Johnson, B. Wardman, S. Sarker, A. Kpravelos, T. Bao, R. Wang, Y. Shoshitaishvili, A. Doupé, and G.-J. Ahn. CrawlPhish: Large-scale Analysis of Client-side Cloaking Techniques in Phishing. In IEEE Symposium on Security and Privacy (S&P), 2021.

[134] Y. Zhang, J. I. Hong, and L. F. Cranor. Cantina: a content-based approach to detecting phishing web sites. In Proceedings of the 16th international conference on World Wide Web, pages 639–648. ACM, 2007.

[135] M. Zhao, B. An, and C. Kiekintveld. Optimizing personalized email filtering thresholds to mitigate sequential spear phishing attacks. In AAAI, pages 658–665, 2016.

Appendix A

Detecting Credential Spearphishing Attacks

A.1 Feature Vectors and Comparators per Sub-Detector

Tables A.1, A.2, A.3 display the features that each of our spearphishing subdetectors use (Chapter 3), along with the comparators that DAS uses for each feature to compute an event's anomaly scores. Each comparator reflects the monotonic nature of each numeric feature: a comparator of "$\leq$" expresses that we expect smaller feature values to indicate a more suspicious event, whereas a comparator of "$\geq$" indicates that larger values correspond to more suspicious events. In each of these tables the "Host" of a URL refers to the full-qualified domain name (FDQN) of the URL.

A.2 Preventative Interstitials

In Section 3.7 we discussed how to extend our spearphishing detector from a real-time alert system to a preventative defense by rewriting suspicious URLs in emails to redirect to an interstitial page. However, this defense will only be practical if employees rarely encounter (false positive) interstitial warning pages; otherwise, warning fatigue will likely cause many users to disregard the warning altogether. To assess this concern, we ran our detector on our entire evaluation dataset (Sep 1, 2013 – Jan 14, 2017) with an average daily budget of 10 alerts, and selected the alerts that fell within our cumulative budget for that window (i.e., selecting the top $B = 10 \times N_{DaysInEvalWindow} = 12,310$ most suspicious click-in-email events). For each recipient (`RCPT TO` email address) that received an email flagged in at least one of those 12,310 alerts, we computed the total number alerts that the recipient received over the entire evaluation time window. Figures A.1 and A.2 show these results in histogram and CDF form. From these figures, we see that roughly half of all employees would only encounter 1 interstitial and over 95% of employees would see fewer than 10 interstitials across the entire time span of nearly 3.5 years. This low volume suggests that organizations could viably deploy the preventative interstitial defense we proposed in practice.

***Name spoofer* Features**	**Comparator for DAS**
Host (FQDN) age of clicked URL (email ts − domain's 1st visit ts)	$\leq$
# visits to clicked URL's host prior to email ts	$\leq$
# weeks that `From` name has sent email on ≥ 5 days	$\geq$
# days that `From` name and `From` addr have appeared together in emails	$\leq$

Table A.1: Summary of the feature vector for our *name spoofer* sub-detector and the "suspiciousness" comparator we provide to DAS for each feature (Chapter 3).

***Previously unseen attacker* Features**	**Comparator for DAS**
Host (FDQN) age of clicked URL (email ts − domain's 1st visit ts)	$\leq$
# visits to clicked URL's host prior to email ts	$\leq$
# days that `From` name has sent email	$\leq$
# days that `From` addr has sent email	$\leq$

Table A.2: Summary of the feature vector for our *previously unseen attacker* sub-detector and the "suspiciousness" comparator we provide to DAS for each feature (Chapter 3).

***Lateral attacker* Features**	**Comparator for DAS**
Host (FQDN) age of clicked URL (email ts − domain's 1st visit ts)	$\leq$
# visits to clicked URL's host prior to email ts	$\leq$
# distinct employees who have previously logged in from the same city as the session's new IP addr	$\leq$
# previous logins by the current employee from the same city as the session's new IP addr	$\leq$

Table A.3: Summary of the feature vector for our *lateral attacker* sub-detector and the "suspiciousness" comparator we provide to DAS for each feature (Chapter 3).

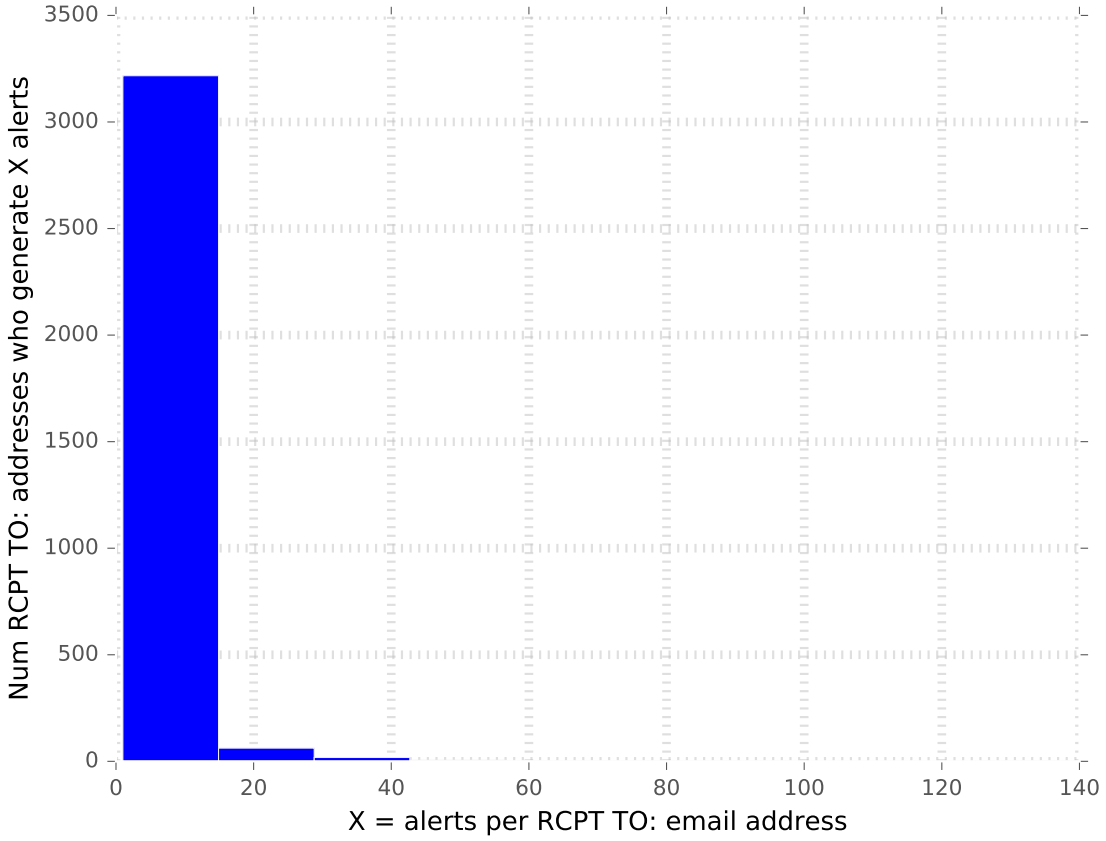

Figure A.1: Histogram of alerts per `RCPT TO` address for our spearphishing detector, under an average budget of 10 alerts per day across the Sep 1, 2013 – Jan 14, 2017 timeframe (Chapter 3).

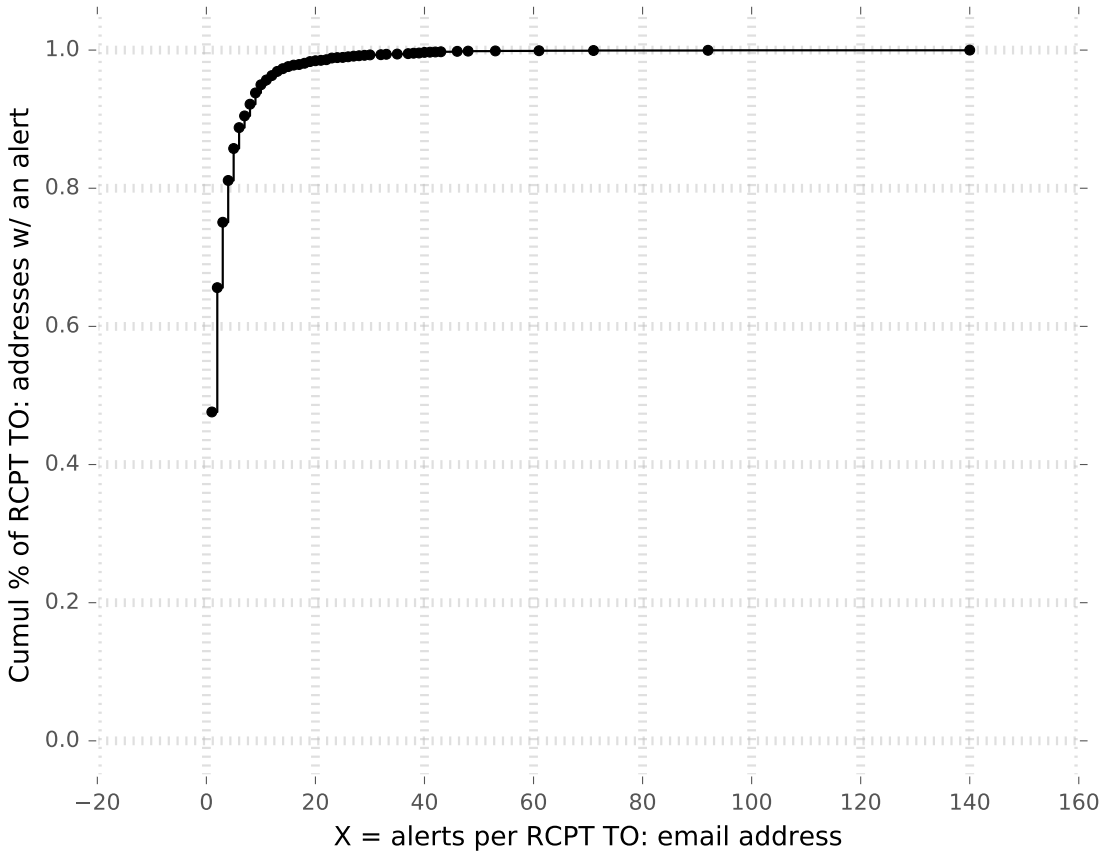

Figure A.2: CDF of alerts per `RCPT TO` address for our spearphishing detector, under an average budget of 10 alerts per day across the Sep 1, 2013 – Jan 14, 2017 timeframe (Chapter 3).

Appendix B

Detecting and Characterizing Lateral Phishing

B.1 Detector Implementation and Evaluation Details

Labeling Phishing Emails

Labeling email as phishing or benign: When manually labeling an email, we started by examining five pieces of information: whether the email was a reported phishing incident, the message content, the suspicious URL that was flagged and if its domain made sense in context, the email's recipients, and the sender. With the exception of a few incidents, we could easily identify a phishing email from the above steps. For example: an email about a "shared Office 365 document" sent to hundreds of unrelated recipients and whose document link pointed to a bit.ly shortened (non-Microsoft) domain; or an email describing an "account security problem" sent by a non-IT employee, where the "account reset" URL pointed to an unrelated domain. For the more difficult cases, we analyzed all replies and forwards in the email chain, and labeled the email as phishing if it either received multiple replies / forwards that expressed alarm or suspicious, or if the hijacked account eventually sent a reply saying that they did not send the phishing email. Finally, as described in Section 5.3, we visited the non-side-effect, suspicious URLs from a sample of the labeled phishing emails. All of the URLs we visited led to either an interstitial warning page (e.g., Google SafeBrowsing), or a spoofed log-on page. For the emails flagged by our detector, but which appeared benign based on examining all the above information, we conservatively labeled them as false positives. In many cases, false positives were readily apparent; e.g., emails where the "suspicious URL" flagged by our detector occurred in the sender's signature and linked to their personal website.

Training exercises vs. actual phishing emails: In addition to distinguishing between a false positive and an attack, we checked to ensure that our lateral phishing incidents represented actual attacks, and not training exercises. First, based on the lateral phishing emails' headers, we verified that all of the sending accounts were legitimate enterprise accounts. Second, all but five of the

attack accounts sent one or more unrelated-to-phishing emails in the preceding month. These two points gave us confidence that the phishing emails came from existing, legitimate accounts, and thus represented actual attacks; i.e., training exercises do not typically hijack an existing account, due to the potential reputation harm this could incur (and enterprise security teams we've previously engaged with do not do this). Furthermore, none of our dataset's lateral phishing incidents are training exercises known to Barracuda, and none of the lateral phishing URLs used domains of known security companies.

Model Tuning and Hyperparameters

Most machine learning models, including Random Forest, require the user to set various hyperparameters that govern the model's training process. To determine the optimal set of hyperparameters for our classifier, we followed machine learning best practices by conducting a three-fold cross-validation grid search over all combinations of the hyperparameters listed below [6].

1. Number of trees: 50 – 500, in steps of 50 (i.e., 50, 100, 150, . . . , 450, 500)
2. Maximum tree depth: 10 – 100, in steps of 10
3. Minimum leaf size: 1, 2, 4, 8
4. Downsampling ratio of (benign / attack) emails: 10, 50, 100, 200

Because our training dataset contained only a few dozen incidents, we used three folds to ensure that each fold in the cross-validation contained several attack instances. Our experiments used a Random Forest model with 64 trees, a maximum depth of 8, a minimum leaf size of 4 elements, and a downsampling of 200 benign emails per 1 attack email, since this configuration produced the highest AUC score [40]. But we note that many of the hyperparameter combinations yielded similar results.

B.2 Additional Detection Approaches

In this section, we explore the two "suboptimal" text-based strategies for identifying lateral phishing that we alluded to earlier in Section 5.4. First, we describe the features and classification process for each of these two strategies, and then evaluate their performance using the same time window and evaluation data from Section 5.5. Finally, we assess the performance of a detector that combines both these two additional strategies with the main strategy presented in Section 5.4 and the overlap in detection across all three approaches.

Design: Fuzzy Phish Matching Detector

The "fuzzy phish matching" detection strategy follows a natural intuition: if the text in a new email closely matches the text in other known phishing emails, then the new email is also probably a

phishing attack. In the context of the lure-exploit framework of phishing attacks [49], this approach identifies phishing emails by looking for thematic lures that previously appeared, and continue to recur, in new phishing attacks.

Features: As input, this detector takes a set of known phishing emails. Given a new email, the detector extracts a similarity feature, the email's *fuzzy phish similarity score*, which measures the similarity between the new email's text and each known phishing email.

For each email, the detector normalizes the message's text by removing signatures and other auto-generated text with the Talon [70] library, lowercasing all words, and removing punctuation. Subsequently, the detector tokenizes the normalized text and converts it into a set of 3-grams of consecutive words. To compute the similarity between two emails, we use the Jaccard similarity between these two sets of 3-grams. After computing the pairwise similarity of the new email's text with all known phishing emails, the detector assigns the highest similarity value as the email's *fuzzy phish similarity score*.

To ensure a low volume of false positives, we then combine this message-content feature with an additional feature that characterizes whether an email contains a potentially dangerous action. Specifically, for each email, we extract a *global URL reputation* feature that quantifies the rarest URL an email contains (using the same procedure discussed in Section 5.4).

Classification: Given a new email, we extract its *fuzzy phish similarity score* and *global URL reputation* features. Since our focus is on URL-based phishing, if an email contains no URLs, we classify it as benign. Otherwise, we apply a set of simple, conservative thresholds to these two features: if an email's text is over 50% similar to a known phish (its *fuzzy phish similarity score* $\geq$ 50%) and it contains a domain whose global ranking is outside of the top 100,000 domains, then we classify the email as phishing; otherwise, the detector labels it as benign.

Design: Template Matching Detector

In a similar spirit to our previous strategy, our Template Detector also tries to identify emails whose message content has a proclivity for phishing usage. However, unlike the prior strategy, which needed a set of historical phishing emails to recognize phishing text, our Template Detector attempts to automatically infer potential phishing content from a large corpus of mostly benign emails. Since phishing emails frequently attempt to masquerade as a legitimate user or service, this second detection approach attempts to build a set of *template texts*: "popular" texts that users frequently encounter and associate with benign services. For example, several attacks in our initial sample of lateral phishing emails contained messages that appeared nearly identical to legitimate emails from the popular Docusign service.

Features: We find these template texts by taking the past month's emails and then mapping each email to a tuple consisting of the email sender's domain and the alphabetically-ordered set of registered domains for all URLs embedded within the email (we call this alphabetically-ordered set the *domain group* of an email). Next, we keep only emails where both the email sender's domain and every domain in its domain group rank in the top 100,000 domains; we also remove any email if the email sender's domain belongs to a popular personal-email provider (e.g., Gmail,

AOL, Comcast, etc.). Additionally, we check that this (sender email domain, domain group) tuple appears in at least 50 received emails per organization for at least 10% of our organizations. These requirements help ensure that we have a set of "popular" emails, in that a reputable sender sent the email, all of its URLs belong to popular domains, and many different organizations receive this type of email. Finally, to extract a set of "popular texts", we group all emails by their (sender email domain, domain group) value, and select the email whose text has the highest 3-gram Jaccard similarity across all emails in the group. Quantitatively, in this final step, for each email, we compute its 3-gram Jaccard similarity with every other email in the group. We then sum up these similarity scores for each email, and select the email in each group with the highest summed score. We refer to this resulting set of emails as a set of *templates*.

Now that we have a set of templates, we can mimic the prior strategy. First, we compute a *template similarity score*, which measures a new email's similarity to any known template, by extracting 3-grams of consecutive words from the new email, computing the Jaccard similarity of the email's 3-grams with every template's 3-grams, and selecting the highest similarity score. Finally, we extract the new email's *global URL reputation* feature using the same procedure as before.

Classification: To classify a new email as lateral phishing or not, we apply the same thresholds to this detector's two features as we did in our Fuzzy Phish Detector: if a new email's text is over 50% similar to any template, and the email contains a domain ranked outside of the top 100,000 domains, our detector labels this new email as an attack; otherwise, it treats the email as benign.

Evaluation: Fuzzy Phish Detector

To evaluate our Fuzzy Phish Detector, we aggregated all emails from *user-reported* lateral phishing incidents, and then used this set of phishing emails as our set of known-phishing emails for extracting a new email's *fuzzy phish similarity score* (Section B.2). To ensure temporal accuracy (i.e., not using phish from the future to detect present phish), we only compared a new email against known-phishing emails sent at least 24 hours prior.

Training and Tuning: On our entire training dataset (Section 5.5), our Fuzzy Phish Detector generated alerts for 2 lateral phishing incidents (which were reported by users) and produces 0 false positives.

Despite generating no false alarms, this approach missed 38 user-reported incidents. These false negatives stem from two causes. First, most known attacks in our dataset use very short email messages, often consisting of a few words with an embedded phishing URL (e.g., "New contract... View Here"), followed by the hijacked user's signature. Although we used several techniques to remove user signatures during our text similarity scoring, at our scale of tens-of-millions of emails, we encountered many email signatures which our set of techniques fail to remove. This failure to remove signatures occurred for several of our known phishing emails, causing them to generate poor text similarity scores with new phishing emails; for short-message phishing emails, the bulk of the text consisted of the hijacked user's signature. Second, very few phishing emails actually used the same or similar text. For example, among phishing emails with short messages,

we observed many iterations of conceptually similar text that use different wording (e.g., "New contract" vs. "Alice shared Document X with you.")

Detection Results: Turning to our test dataset (Section 5.5), this approach produced alerts for 12 incidents. Of these, 8 are in fact lateral phishing; the remaining 4 incidents are false positives. For the same reasons we saw in our training dataset, this detector exhibited a high false negative rate, missing 57 user-reported incidents. Nonetheless, despite this strategy's high false negative rate, we found that it generated virtually no false positives across a test dataset of tens-of-millions of emails. Moreover, among the 8 lateral phishing incidents it detected, 4 incidents were not reported by a user.

Evaluation: Template Detector

Training and Tuning: Prior to extracting features or classifying a new email, our Template Detector uses the past month's emails (across all of our training organizations) to generate a set of templates. Then, given a new email, this approach extracts the email's features and classifies it as described earlier in Appendix B.2.

Running this approach on our training dataset, we find that this strategy correctly flagged 4 incidents as lateral phishing (where 2 incidents do not appear to be reported by a user). It generated only one additional alert, which turned out to be a phishing training exercise email. In all of these cases, the attackers (and simulation) closely mimicked a legitimate Docusign email's content, but replaced the main "shared document" link with a phishing URL.

Examining the 38 user-reported incidents it missed, we found that the Template Detector is simply ill-suited for identifying the majority of lateral phishing emails: the text of the attacks it missed simply does not appear to mimic any popular, real email. For example, for many of the training dataset attacks it failed to detect, the phishing emails often presented only a short message such as "Please see attached" or "Here is the new document" (in addition to the hijacked user's signature).

Detection Results: Across our test dataset, our Template Detector generated alerts for only 8 total incidents, all of which are phishing emails that come from external sources who spoof a fake username at the victim organization. Mirroring our training dataset findings, all of the user-reported lateral phishing incidents in our test dataset contain phishing messages which do not closely match a legitimate, popular email's text. As such, based on the underlying motivations and assumptions for our Template Detector, this approach will have a difficult time detecting these kinds of attacks. Nonetheless, as with our training dataset, this detection strategy produced 0 false positives across tens of millions of emails, while still flagging several phishing incidents (albeit ones generated by external spoofing).

Combined Detector

We can combine our three detection strategies (including our main approach Section 5.4) into one detector by labeling a new email as lateral phishing if any of the techniques deems it phishing. Throughout this section, we also refer to each of the strategies as a "subdetector".

Combined Detection Results: On the entire test dataset, this aggregate detector achieved a Recall of 87.3%, a Precision of 23.3%, and a False Positive Rate of 0.00036% (one false positive per 277,000 employee-sent emails). Although its precision is lower than desired, the overall low volume of false alarms it generates could enable this detector to be operationally viable.

Our test dataset consists of 52 organizations from our training dataset, plus a held-out set of 40 new organizations; we find that our detector performed comparably on both. Breaking down our test dataset performance numbers, our detector achieved a recall of 81.8% and a precision of 24.8% across our training organizations versus a recall of 91.0% and a precision of 22.8% for the held-out orgs.

Detector overlap: Of the 97 test dataset incidents found by our aggregate detector, 90 are detected by only one subdetector, and 7 incidents are detected by two subdetectors. For all of the 7 incidents, the two overlapping subdetectors that detected them are the Fuzzy Phish Detector and the primary detection strategy (Section 5.4). Of the remaining 90 incidents, the primary detection strategy discussed in Chapter 5 detects all-but-one incident, with the final one coming from the Fuzzy Phish Detector. As we explored earlier in Appendix B.2, this result reflects the fact that the text of phishing emails exhibits frequent churn over time, causing our two text-similarity driven strategies to miss new attacks that our main approach detects.

B.3 Lateral Phishing: Additional Temporal Dynamics

In this section, we characterize several additional timing-related aspects of the attacks in our lateral phishing dataset. Taken together, our analysis suggests that the lateral phishing we studied reflected a mix of automated and manual attacks, where some attackers leveraged their hijacked accounts in a manual fashion, and others appeared to use automated tools to facilitate sending phishing emails at scale.

Duration of compromise per account: Figure B.1 shows how much time elapses between the first lateral phishing email and the last phishing email sent per hijacked account. From this figure, we see that 54 of the hijacked accounts send only one phishing email, or manage to send all of their phish with the same sent timestamp (e.g., by using an automated phishing kit that sends a large burst of emails at the same time). On the opposite extreme, we see one account where over 20 days elapse between their first and last phishing email; in this case, the account appears to have been re-compromised since this account's first few phishing emails all share the same subject and rare domain, but later on the account's final phishing emails use a completely different subject and rare domain. Overall, nearly 90% of attackers in our dataset use a hijacked account for under an hour, and do not subsequently use it to send phish (that we know of). However, we see that among

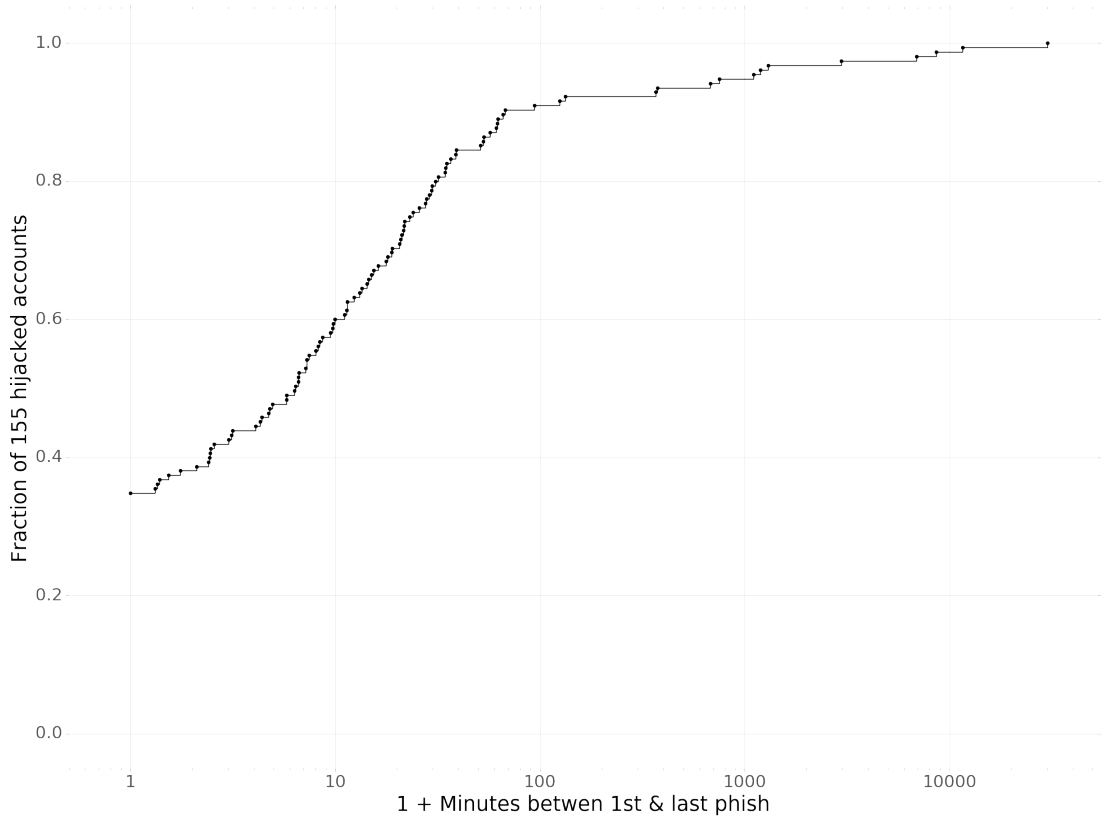

Figure B.1: CDF: fraction of hijacked accounts where X minutes elapsed between the first phishing email they sent to the last phishing email they sent. All values are incremented by 1 to facilitate log-scaling (i.e., from the graph, we see that all of the phish from roughly 33% of senders share the same sent timestamp; either because the attacker only sent one phishing email, or potentially because of the use of an automated script that sends multiple phishing emails all at the same time.)

the remaining 10% (15 hijacked accounts), several of them casually send phishing emails from the compromised account over a span of several days (with the longest being an 8-day separation between the attacker's first and last phishing email).

Delay between an attacker's phishing emails: To characterize whether any attackers appear to have employed automated emailing tools, we studied the interarrival properties between emails that belong to the same incident. Namely, how much time elapsed between phishing emails with the same subject and from the same compromised account? First, we removed all incidents that generate only one email: 52 out of our total 180 incidents have this property. Examining the remaining incidents, Figure B.2 illustrates the minimum, median, and maximum interarrival times between consecutive phishing emails sent by the same hijacked account (e.g., studying the black (uppermost) curve, we can see that about 10% of hijacked accounts send at least two emails within one second of each other); Figure B.3 illustrates the relationship between how many seconds elapsed between consecutive phishing emails in an incident, and the total number of phishing emails in

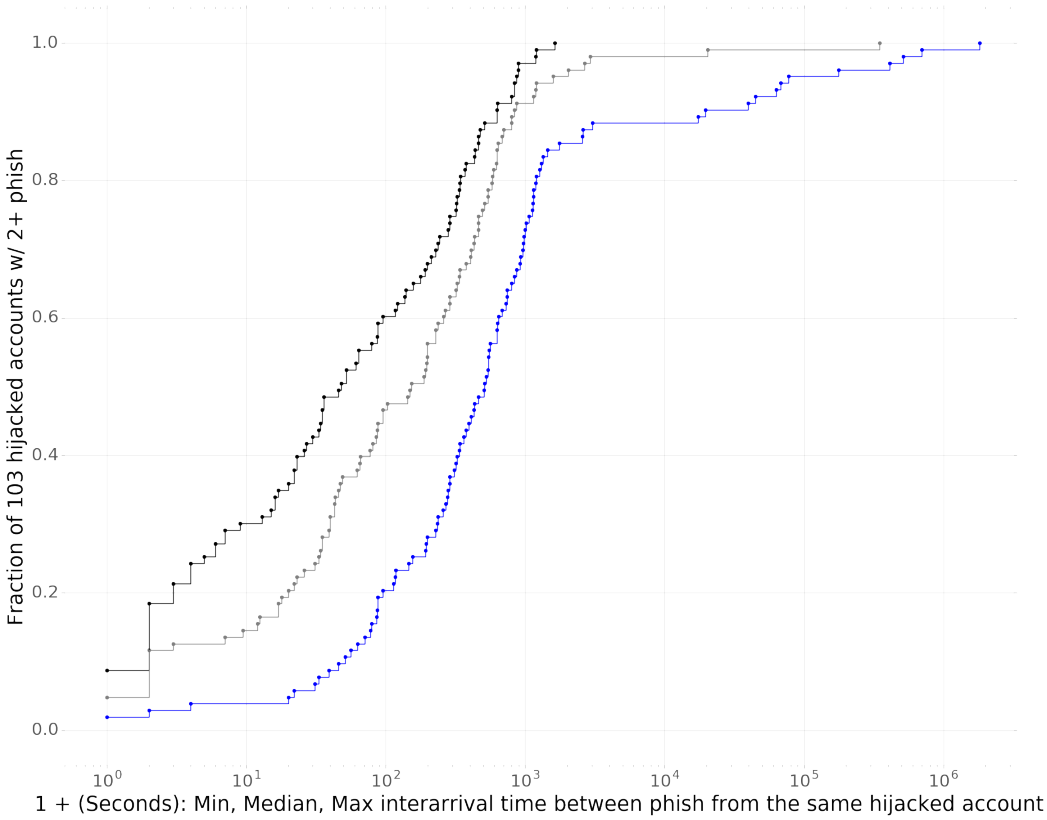

Figure B.2: CDF: fraction of multi-email incidents with a min, median, and max interarrival time (between consecutive phishing emails) of X seconds. The black curve (uppermost) corresponds to the min interarrival time, the gray (middle) curve corresponds to the median, and the blue (bottom) curve corresponds to the max.

an incident. From these plots, we see that 13 hijacked accounts sent at least 2 phishing emails within 2 seconds of each other, suggesting that for these attacks, the adversary might have used some kind of automated tool or phishing kit. In particular, while all emails from these hijacked accounts shared the same subject, each email had a distinct set of recipients. Additionally, two of these hijacked accounts sent over 100 phish emails, where the body text was unique for each recipient (greeting the recipient by name), which strongly suggests the attacker leveraged some form of automation.

In contrast, we observed that for 49 hijacked accounts, the *minimum* interarrival between any two phish they sent was at least one minute, suggesting that the adversary might have manually crafted and sent their phishing emails. As with the potential automation-leveraging attackers, the phishing emails sent from these hijacked accounts each have distinct recipient sets. However unlike the earlier set of attackers, the phishing emails emanating from 15 of these 'slow' hijacked accounts used 2–3 different subjects across their attacks. As we discussed earlier in Section 5.6, these subjects often differed only by a few characters, which suggests that they may indeed be hand-crafted by the attacker (as opposed to generated via automation).

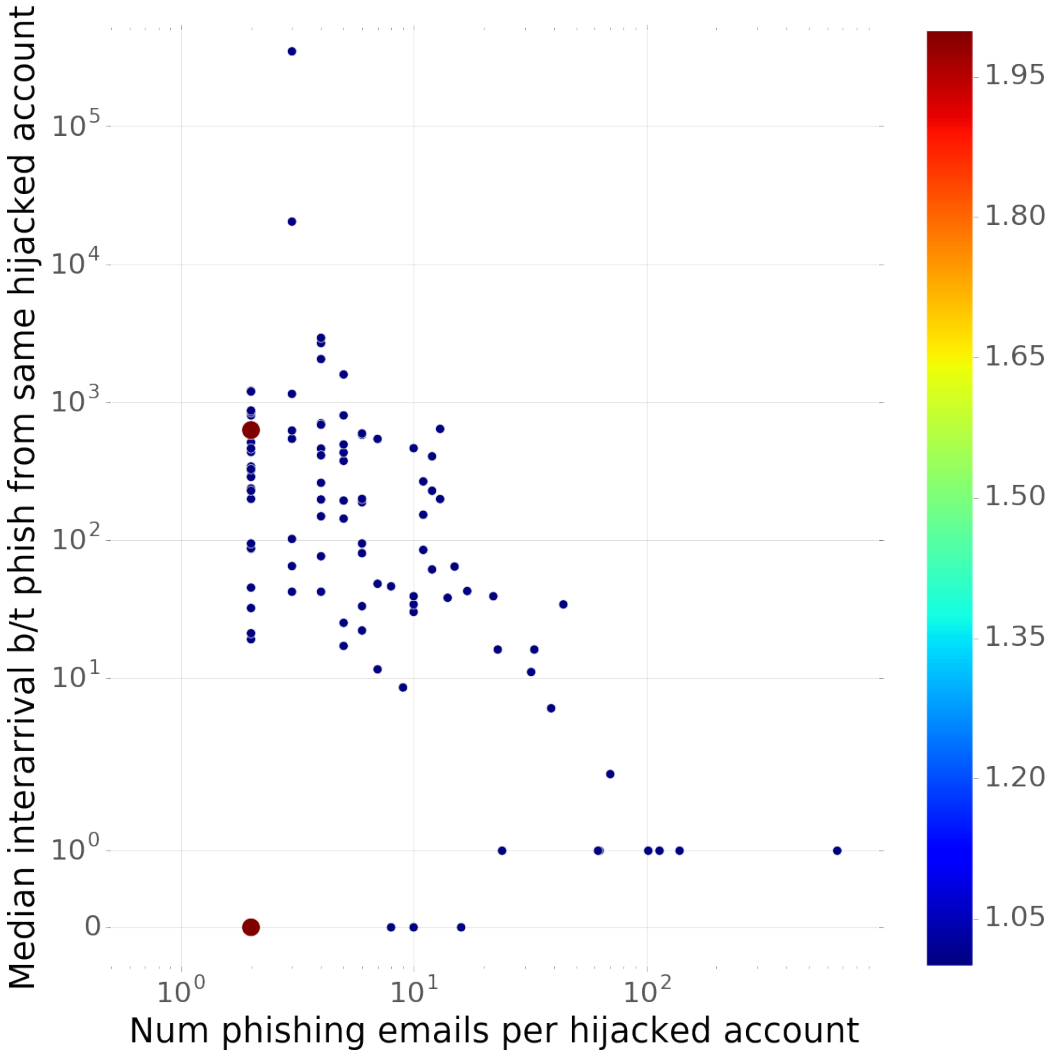

Figure B.3: Scaled scatter plot: each point represents a hijacked account, where the x-coordinate reflects the total number of phishing emails that the account sent, and where the y-coordinate shows the median interarrival time (in seconds) between consecutive attack emails from the same account. The size and color of a point is used to indicate if multiple incidents share the same x and y coordinates

Finally, with respect to the hijacked accounts with long delays between the attacks they send, one account with a nearly two weeks between one phishing email and the subsequent one. These two phishing emails used different phishing URLs, subjects, and body messages, suggesting that this account may have been re-compromised in the intervening timeframe.

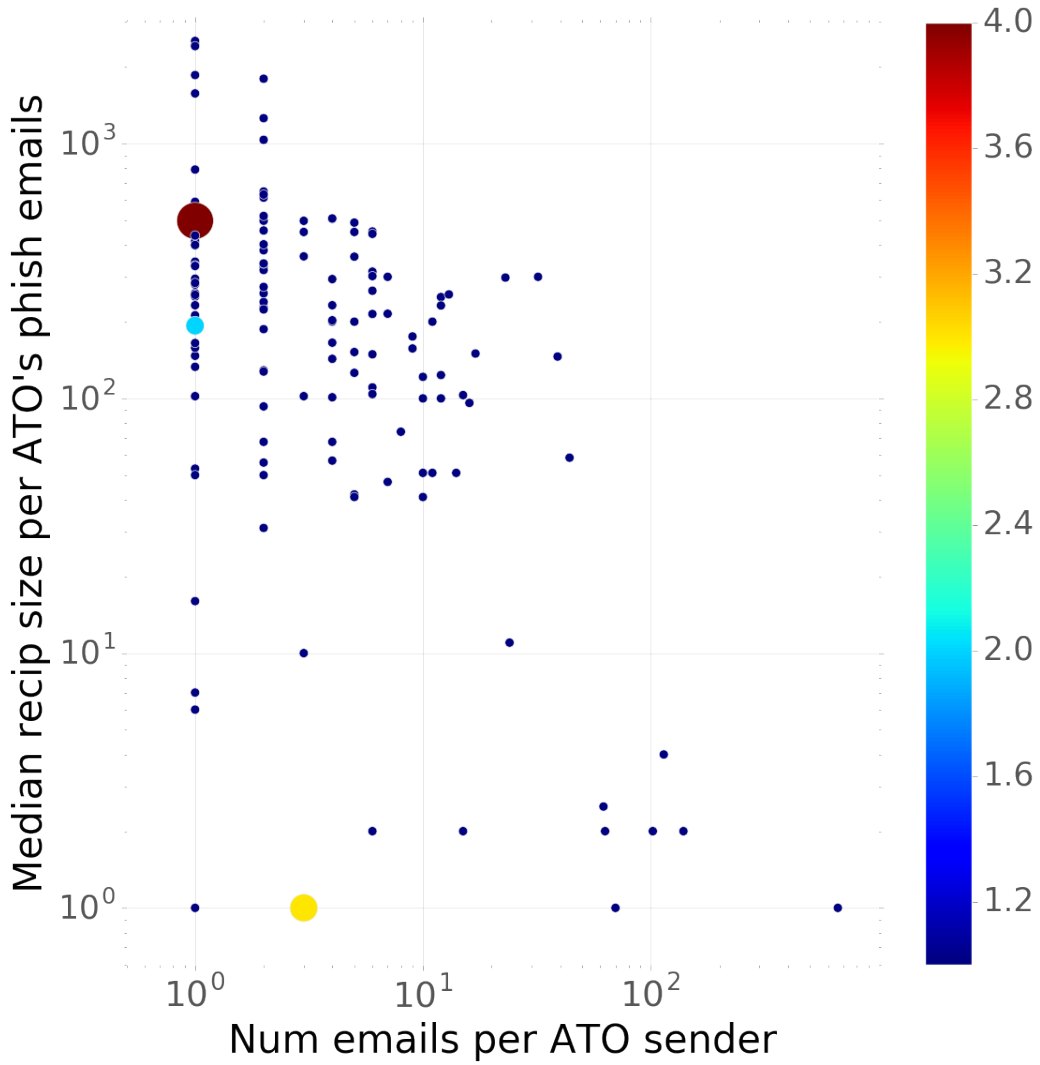

Figure B.4: Scatterplot of the median # of recipients across all lateral phish sent by a hijacked account vs. the total number of phishing emails sent by the hijacked account.

B.4 Exploits Used in Lateral Phishing

Our work in Chapter 5 focused on lateral phishing emails that embedded a malicious URL in their email message. However, as noted in Section 5.3, our dataset contained 12 confirmed, user-reported incidents of lateral phishing that employed a malicious attachment; these incidents each came from a different hijacked user and collectively account for 40 of our lateral phishing emails. These attachment-based attacks shared many of the same characteristics as the URL-based phishing emails in our dataset. For example, in total, these attachment-driven attacks contacted 8,092 distinct recipients, of which 4,092 (50.6%) were fellow employees. Additionally, of the 12 hijacked

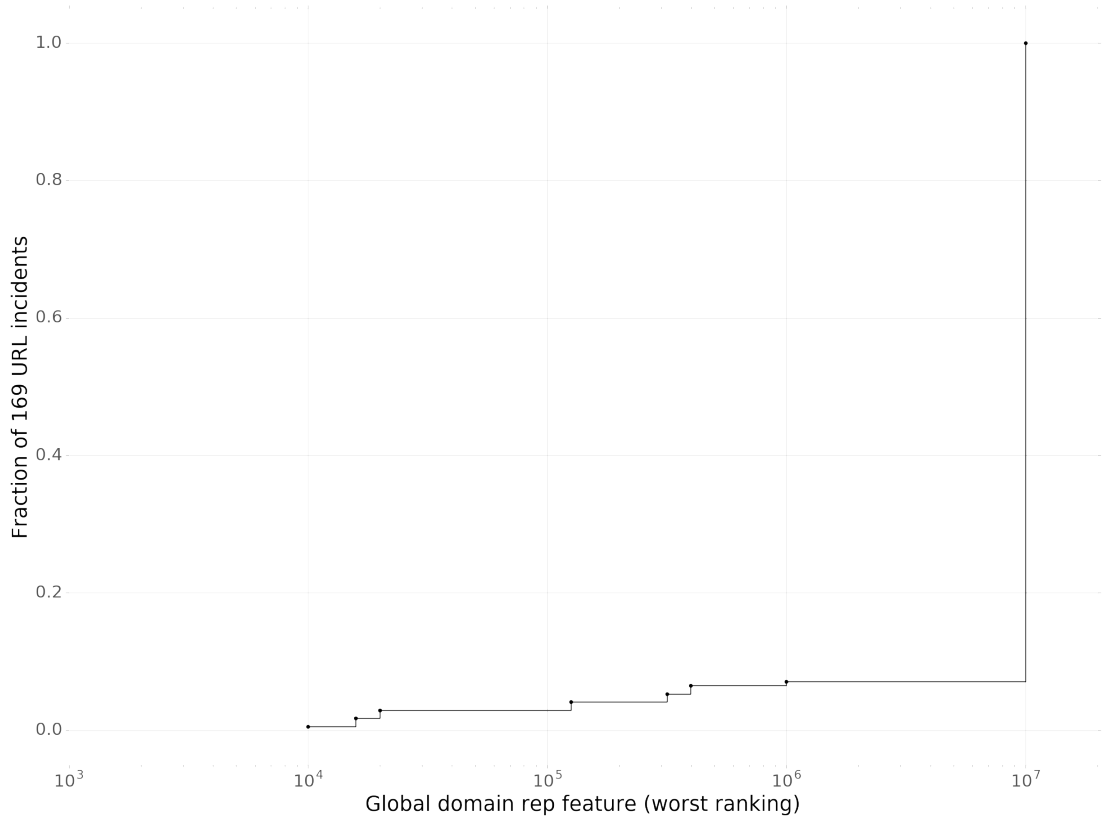

Figure B.5: CDF of the fraction of incidents that contained a malicious URL, where the malicious URL had a global ranking value (Section 5.4) of X (logscaled). Domain ranking values are rounded to the nearest 5000, and unranked domains are assigned a default value of 10 million (X = 7).

accounts, all-but-one of the accounts sent phishing emails to over 100 recipients total. While our dataset does not contain the attachments in these phishing emails, we found that 9 out of the 12 contained fairly generic lures (e.g., "please see attached"); the remaining few appeared to use more tailored, organization-specific lures (e.g., one purported to be an attachment related to a company anniversary event).

Turning our attention to phishing emails that use a malicious URL, Figure B.5 shows the distribution of the global domain ranking Section 5.4 for each incident's phishing URL (rounded to the nearest 5000). Recall that this global value is a domain's rank on the Cisco Umbrella's Top 1-Million domain list; and unranked domains, URLs on content hosting sites, and URLs on link-shortening domains that fail to resolve are assigned a dummy value of 10 million. From this graph, we see that all but 164 incidents (97% of those that use a malicious) hosted their phishing URL on either a domain outside of the top 100,000, or a content hosting or shortening domain. Of these 164 incidents, 8 used domain shortening services, and another 7 linked to a popular content hosting site. Analyzing the remaining 5 incidents whose URLs reside on higher ranked domains, three of them used a shortening or link-redirection service not on our list of shortening domains; one of

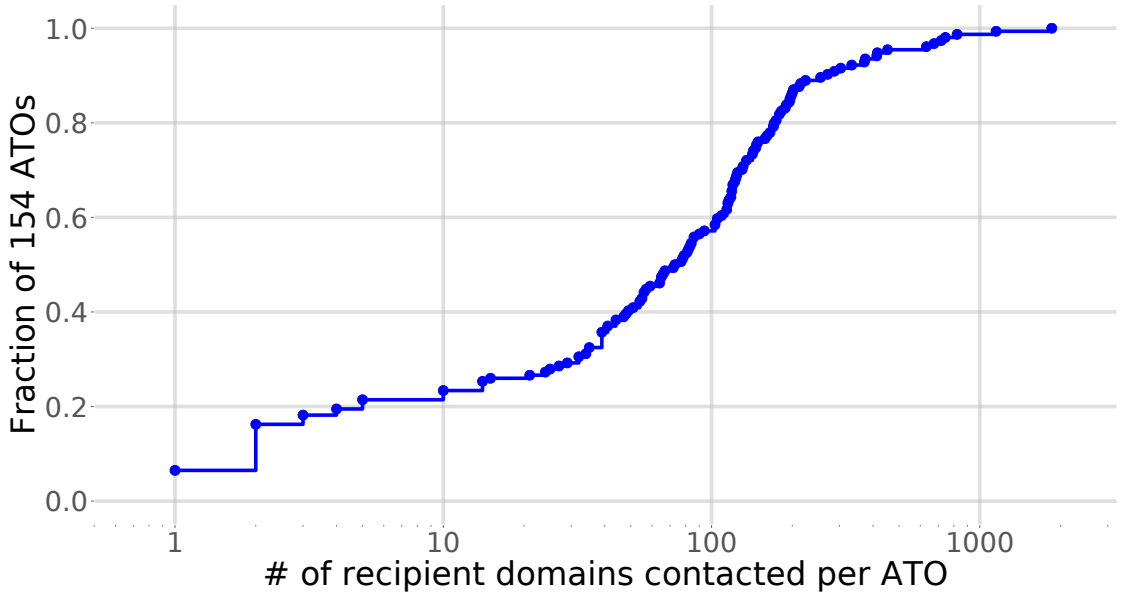

Figure B.6: CDF of the fraction of lateral phishers who send attacks to x distinct recipient email domains. (Section 5.6)

them linked to a content hosting service we were unaware of; and the final one used a popular form building/hosting service.

B.5 Additional Figures

Reach of lateral phishing attacks: Figure B.6 shows an approximation of the number of different organizations (recipient address domains) targeted by the attackers in our dataset. Nearly 80% of lateral phishers in our dataset send their attacks to recipients at 10 or more organizations.

Language of lateral phishing messages: Figure B.7 shows the distribution for how often a word appeared across our dataset's lateral phishing incidents. Earlier in Section 5.6, Table 5.5 showed the frequencies for the top 10 words across lateral phishing incidents.

Distribution of organizations by sampling method: Figures B.8 and B.9 show further details about the organizations in our dataset, extending the characterization in Section 5.3. In particular, Figure B.8 shows the economic sector and Figure B.9 shows the size distributions of our dataset's organizations by sampling method (i.e., the organizations we sampled from those with reported lateral phishing versus those we sampled from all organizations).

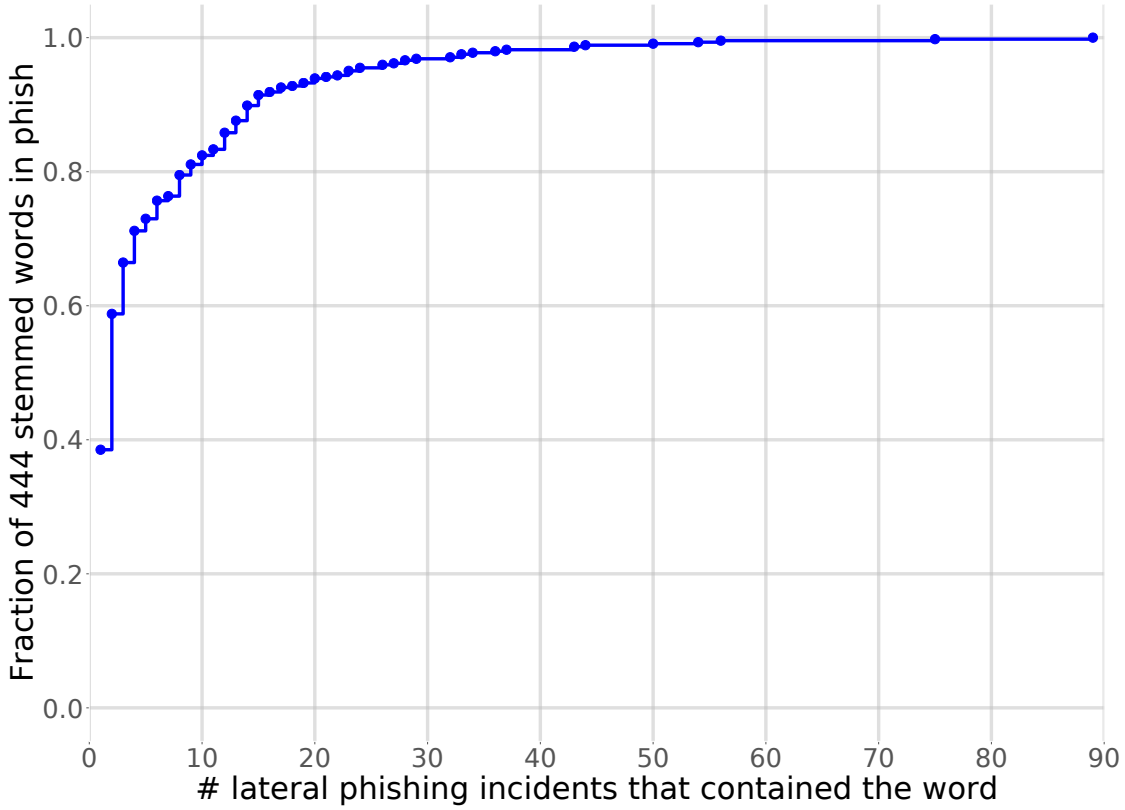

Figure B.7: CDF of the fraction of common, stemmed English words that appear in x lateral phishing incidents. (Section 5.6)

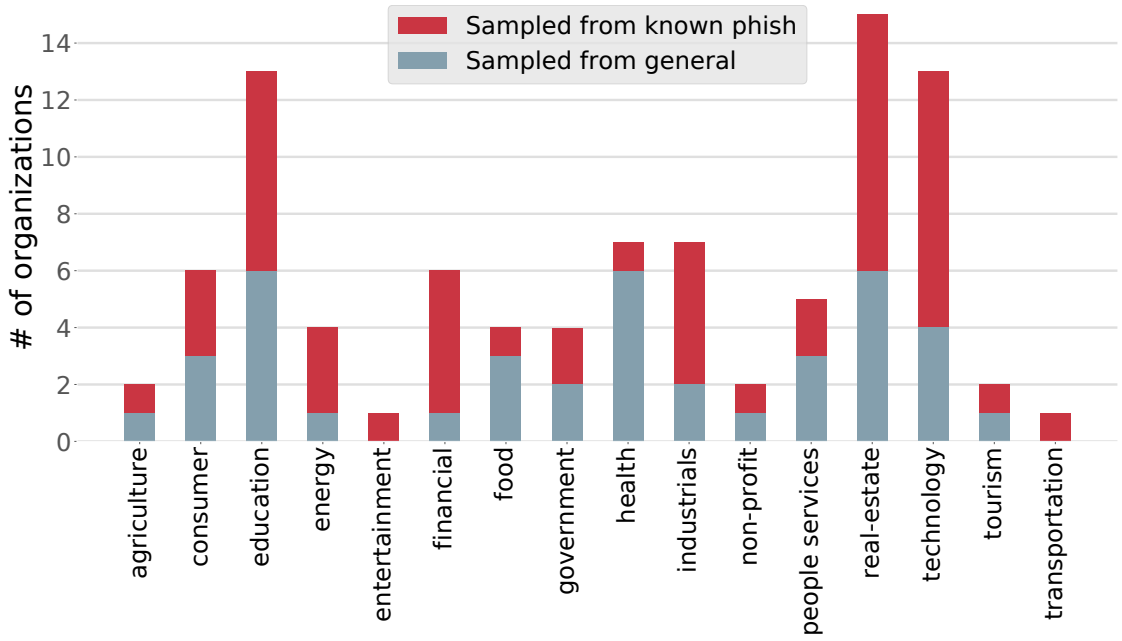

Figure B.8: Distribution of our dataset's organizations by economic sector: organizations that came from sampling the pool of organizations with reported phishing versus the ones that came from sampling the pool of all organizations.

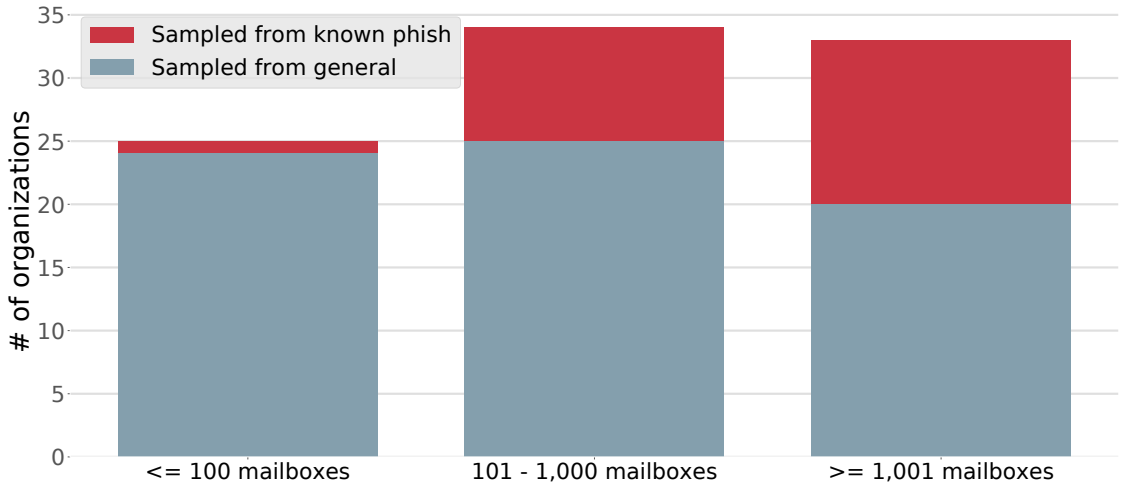

Figure B.9: Distribution of our dataset's organizations by size (number of accounts): organizations that came from sampling the pool of organizations with reported phishing versus the ones that came from sampling the pool of all organizations.

Appendix C

Detecting Lateral Movement Attacks

C.1 Filtering Spurious Logins

Filtering Windows logins:

As noted in prior work [59], many "logins" between internal machines in Windows enterprise environments do not actually represent a meaningful remote access event. Rather, these login records often correspond to uninteresting artifacts and special API calls that result from Windows enterprise logging, and do not provide a user with the ability to access data or execute remote commands on the destination machine. Filtering out these logins from our data results in a 40x reduction, which comes primarily from removing three types of logins: printing jobs, authentications into update and logging servers, and non-administrator logins to Windows Domain Controllers.

Many organizations, such as Dropbox, centralize services such as user authentication, permission management, and global machine configurations (e.g., forced update schedules and default machine settings) under a small set of specialized servers known as "Domain Controllers". This process, known as joining machines under an enterprise "domain", generates many login records that reflect banal interactions with these domain controllers and correspond to artifacts of the Windows logging process. For example, each of these Domain Controllers act as a Kerberos KDC (key distribution center) and TGS (ticket granting server), and generate multiple "login" events into a domain controller whenever a source machine attempts to access another machine via Kerberos; e.g., requests by a user for Kerberos TGT's and service tickets will generate login events from the user into the domain controller. However, these login events actually correspond to the domain controller authenticating the user's credentials through a limited API call that do not involve any meaningful read, write, or executable access (login) to the domain controller itself. At Dropbox, only a small subset of elevated administrator credentials can perform meaningful operations on a domain controller; as a result, we can easily filter out the more spurious logins by discarding any "login" into a domain controller where the target username does not match one of these elevated credentials (designated by a special username suffix, e.g., "alice-administrator").

Another prevalent class of spurious logins occurs whenever users print a document from one of enterprise printers, which create multiple "login" (authentication) events by the user's machine

into a central networking printing server (who in turn, generates a set of authentication events to verify the machine and user's permissions with the organization's domain controllers). Finally, we filter login events whose destination machine belongs to a set of update and logging servers at Dropbox, which generate login events when enterprise machines periodically contact the servers to check for updates and/or transmit their host monitoring logs (e.g., machine status and audit logs collected for security purposes). In total, after filtering out these three sources of spurious login events, our dataset contains roughly 19.5 million login events.

Filtering automation logins: We further winnow our dataset by removing internal logins that result from automated scripts or processes. Specifically, Hopper analyzes a historical set of logins and identifies all login edges, a triplet of (source, destination, and username), that (1) occurs frequently across our dataset[1], (2) occur on at least 50% of the historical days, and (3) have a target username that does *not* match any employee's account (i.e., a non-human username). Hopper then outputs a list of these edges as candidates for automation related logins. An organization's security team can then assess these logins, and approve all or a subset of them. Hopper then removes any login whose edge (exact source, destination, and target user) match an edge listed in this automation set.

In our dataset, Hopper produces a set of approximately 30 automation edges that account for over 16 million login events. Based on manually inspecting these automation logins, these logins reflected mundane operations with minimally privileged service accounts via a restricted set of remote-API calls (e.g., specific remctl calls [2] exposed by the destination machines). For example, many of these logins resulted from file synchronization operations between a central "leader" node and geographic replicas (e.g., a central software repository machine syncing its content with replicated, regional servers). Another common category of these automation logins correspond to version controller and bug tracking software performing git operations to synchronize state among each other; these internal logins occurred under a restricted "git" user account that have access to a limited API of git operations.

C.2 Benign Movement Scenarios

When developing and analyzing the alerts produced by Hopper on the first three months of our data (Jan 1, 2019 to Apr 1, 2019), we observed three common benign reasons for paths where Hopper infers a possible credential switch from its causal user. We developed a set of heuristics for identifying if a path falls under one of these scenarios and do not generate an alert if so. In total, these three cases label approximately 170,000 paths as benign (out of the approximately 10M paths produced by Hopper's causality engine).

The first class of benign paths correspond to one-hop paths (logins) that fall under three subcategories: logins by a new user, a new machine, and/or logins that relate to machine provisioning. Logins by a new user or new machine sometimes cause Hopper to infer a mismatch between the

[1]In our work, we define a frequently occurring edge as one that occurs greater than $N = 24 \times D$ times, where D equals the number of days in the historical dataset (i.e., in total, the edge occurs at least as often as a process that runs once every hour on each day in our the historical dataset).

login's causal user and the login's target user if an organization's inventory database has not yet relabeled the new employee as the machine's owner. We compute the age of a user and machine (i.e., the difference between the current login and the first occurrence of a user/machine in an organization's inventory database and logs), and suppress any alerts for users or machines less than one week old. Logins related to machine provisioning, when a system administrator re-images and configures a machine to reset it and/or reassign the machine to a new owner, also produced many one-hop paths with a clear-credential switch. As part of this process, a system administrator will run a script that authenticates and logins into various specialized servers to configure the machine (e.g., installing the operating system and necessary software from internal servers, configuring the machine's new user and permissions at the organization's Domain Controllers, etc.). Because these logins use the system administrator's credentials, Hopper will infer that a credential switch has occurred because the system administrator (target username) does not equal the source machine's owner (causal user). To identify login events that relate to machine re-provisioning, Hopper checks for three properties (1) the login's destination belongs to a set of machine imaging and provisioning servers within the enterprise, (2) the login's target user corresponds to a system administrator, and (3) the login originates from one of the dedicated subnets used for machine imaging at any of Dropbox's offices (based on domain-provided information about the environment's subnets). Whenever Hopper classifies a login with these three properties, it does not run its causality engine and does not generate an alert. In total, Hopper identifies 125,743 benign paths that correspond to one of these cases of a new machine or (re)provisioning-related logins. Second, the use of (non-human) service accounts produces 42,008 one-hop paths that Hopper would otherwise label as cases of clear-credential switching. In these logins, a legitimate user did perform the login using a set of credentials (target username) that did not match their own; however, these logins correspond to an expected credential switch, as part of accessing a service within the enterprise. For example, one set of these logins includes users running a script to launch test and debugging jobs when building a new version of the organization's desktop application; part of this script includes remote commands issued to the build and test machines under a service account (e.g., user = "test-services"). Hopper infers a set of these service usernames by identifying any username that does not match an employee username and that over ten different source machines have used in successful logins across a set of historical data. To ensure that usernames inferred by Hopper do not provide widespread access or highly privileged capabilities, Hopper outputs the set of inferred service accounts for an organization's security team to confirm, and uses only the set of approved service usernames when filtering logins under this benign scenario. Because these accounts are designed for a limited and specific service operation, organizations can mitigate the risk of lateral movement via these credentials by configuring them with a limited set of permissions to a specific set of machines; at Dropbox, many of these service accounts also access their destinations via a limited remote command API [2], as opposed to a creating full interactive session.

The final benign scenario involves logins to and from a bastion host. Organizations often segment parts of their network for improved efficiency, maintenance, and security by placing a set of machines behind a hardened bastion host [23, 126]. In order to access a server within this network segment, a user must first tunnel and authenticate through the network segment's bastion. Dropbox's corporate network contains a few such network segments, and approximately 2,000

	Exploration	Aggressive Spread	Targeted
No stealth	41	41	40
Active Cred stealth	14	14	13
Prior Edge stealth	41	41	40
Full stealth	14	14	13

Table C.1: Distribution of successful attacks across the scenarios simulated by our attack framework (Section 6.7 and Appendix C.3). In total, our framework generated 326 attack simulations that we used to evaluate the detection rate of Hopper and a prior state-of-the-art detection system.

logins to machines behind a bastion node caused Hopper to generate causal paths with uncertain causality. These logins created confusion in Hopper's causality engine because they occurred close in time with another user's login into a machine within the network segment; since both logins need to traverse through the segment's bastion node and occur close in time, Hopper cannot tell which inbound login into the bastion caused which outbound login from the bastion to the path's final destination. Because bastion machines correspond to hardened hosts, perform a limited set of operations (authentication and connection forwarding), and often do not allow users to establish logins onto the host itself, these paths do not provide an opportunity for a switch in credentials. Thus, given a list of bastion hosts at an organization, Hopper does not alert on any of these one-hop or two-hop paths.

C.3 Synthesizing Realistic Attacks

To supplement our evaluation with additional, realistic attacks, we developed a framework to synthesize lateral movement logins that correspond to one of twelve attack scenarios. Each scenario consists of a pair of parameters, that specify one of three attacker goals and one of four stealthiness levels (described below).

Attack Synthesis Procedure: Given a specific attack scenario and starting "foothold" machine that an attacker has already compromised, our framework selects a random starting time for the attack (during the time period where the foothold machine remains active in our data). Next, our framework iteratively generates a set of attack logins until it reaches a termination condition, specified by the attack scenario's goal.

During each iteration, our framework identifies a set of all possible "next hop" logins; this set corresponds to all combinations of the internal machines an attacker has compromised (potential source machines), all of the credentials the attacker has compromised (potential target usernames), and all the machines (potential destinations) that the attackers' compromised credentials have successfully accessed in any login across our entire dataset. Modeling a powerful adversary, our framework assumes that the attacker has the ability to determine which destinations a compromised credential can access (e.g., via out-of-scope discovery or internal reconnaissance operations, such as Active Directory enumeration [77]). At the start of an attack, the set of compromised ma-

chines and credentials corresponds to just the attacker's foothold machine and the credentials of the foothold machine's owner.

Our framework then prunes this list of potential next hops based on the attack scenario's stealthiness and goal. After pruning this list, our framework randomly selects one of these logins as the next attack login, and sets the time of this login by randomly adding an offset between 0–12 hours after the prior attack login. Our framework marks the new login's destination as compromised as well as the credentials of any user who recently logged into the destination (i.e., simulating an attacker who "compromises" the passwords / credentials of all usernames that have logged into the new destination machine within the past seven days). An attack terminates (stops generating additional logins) if this new login fulfills the scenario's goal (described below), or if the attack has visited every possible destination that its compromised credential set can access.

Attack Scenarios: Our framework's twelve lateral movement scenarios come from pairing an attack goal with an attack "stealthiness" level. An attack's goal specifies when an attack has succeeded, i.e., when our synthesis framework stops generating attack logins. An attack's stealthiness level governs the pruning phase of our attack generation framework, which winnows a candidate set of potential next hops to only those that abide by a certain level of stealthiness.

1. **Attack Goal**

    a) Aggressive Spread: this goal simulates an attacker who attempts to compromise as many machines in the enterprise as possible (e.g., a ransomware attack). When synthesizing these attacks, our framework generates logins in a breadth-first traversal, iterating over each compromised credential the attack has acquired and moving to all destinations accessible by each credential; if our framework has already accessed a destination machine during an earlier step of the attack, with any set of credentials, it does not generate an additional login to the machine. This attack terminates once our framework has either generated a login edge into every possible destination that its compromised credential set can access (in accordance with the scenario's specified stealthiness), or when it exceeds 50 attack logins.

    b) Targeted Compromise: our framework simulates a targeted attack by synthesizing attack hops until it produces a login into one of approximately twenty-five "high-value machines" that we selected; these machines reflect a representative range of machines that many real-world attacks have targeted, such as servers that manage user credentials and permissions (e.g., Windows domain controllers) and critical infrastructure (e.g., an organization's DNS servers).

    Under this attack goal, our framework pre-computes a set of the shortest paths from the initial foothold to any machine with credentials that enable access to one of these high-value machines (e.g., machines that system administrators have logged into). After following one of these paths and compromising a necessary set of credentials, our framework then computes the shortest path from any of the attack's compromised machines to any of the high-value machines. As our framework synthesizes attack logins,

it prunes the set of candidate hops to ensure that the attack follows a shortest path to these two goals.

c) Opportunistic Exploration: in this attack, our framework synthesizes attack logins until it produces a login that accesses a new destination that the initial victim does not have access to. During each step, this attack enumerates a set of viable edge movements (triplets of source machines that the attack has accessed, compromised credentials, and accessible destination machines). Our framework then prunes this set to only include edges that access a new destination machine (not yet visited during the attack), and randomly chooses one of the remaining edges for the next login in the attack path. This scenario helps evaluate Hopper against attacks that might compromise an interesting or sensitive machine that we did not specify in the set of high-value machines for our Targeted Compromise goal.

2. **Attack Stealthiness**

a) No Stealthiness: no pruning performed. To select the next attack login, our framework randomly chooses one of the candidate next hops.

b) Active Credential Use: prunes the set of candidate next hops to only logins that use the credentials of the initial victim or hops that use a new set of credentials if a legitimate user has recently (within the past 24 hours) logged into the hop's source machine; i.e., the attacker only performs a login with a new set of credentials if it creates paths with unclear causality. This scenario simulates a stealthy attacker who only uses credentials from machines where the legitimate user was recently active, enabling them to potentially evade detection because the true user could plausibly have made the attack login.

c) Known Edge Movement: prunes the set of candidate next hops to only ones that traverse graph edges with prior history. This stealthiness simulates an attacker who attempts to evade detection by only moving between machines with prior, benign logins (i.e., creates only non-anomalous attack edges).

d) Combined Stealthiness: applies both the "Active Credential Use" and "Known Edge" stealthiness criteria to prune the set of logins that our framework selects for the attack's next hop.

While not perfectly accurate, this framework provides a good approximation of a powerful and realistic adversary: once an attacker moves to a new destination, our framework assumes that the attacker can exploit a local privilege vulnerability to obtain full administrative privileges on a machine, and thus compromise or hijack the credentials of any user who has logged in recently. Because cached credentials have varying lifetimes based on the login protocol and local configurations of a destination machine, our seven-day window provides a conservative estimate of how long other users' credentials remain vulnerable on a machine. Moreover, although a real-world attacker might not know all of the machines an arbitrary user's credential can access, we assume that

an attacker can identify all of the machines a credential can successfully access when generating the next hop's set of potential destinations. Ideally our framework would know precisely which machines a given credential can access, but our research does not have easy access to this data, so we approximate this set by computing all of the machines a username has ever accessed across our dataset's logins.

Synthesizing Attacks: We randomly selected a set of 50 users from the set of all non-system-administrator employees who had at least one internal login across our final, post-filtered dataset (Section 6.3). For each user in this set of 50 "initial victims", we ran our framework to synthesize all twelve lateral movement scenarios from each initial victim's laptop. i.e., our synthetic attacks simulated an attacker compromising a random employee's laptop, where each of the twelve attack scenarios per starting victim simulates a different type of attacker stealthiness and goal. If an attack's lateral movement failed (i.e., could not acquire and use an a new set of credentials), we re-ran our framework on every day in our evaluation window (where the attack foothold and initial victim still existed at Dropbox). If our framework produced an attack that succeeded on any of these days, we replaced the failed lateral movement with a random one of these sets of successful attack logins

In total, this procedure produced 326 synthetic attacks that successfully conducted lateral movement. For 9 of the initial victims, the attacker had no ability to move laterally, because the user only accessed a small set of servers that either only they or team members with identical access permissions accessed; thus, an attacker starting from these initial victims' machines lacked any path to additional machines. Additionally, for 36 starting victims, attackers lacked a stealthy path that would enable them to conduct lateral movement. Even though one of these starting victims might encounter another user's credentials that would enable new machine access, the other employee never used their credentials to login to a new machine from the mutually accessible server(s). In these situations, an attacker could not use these new credentials without generating a never-before-seen login event; therefore, our synthesis framework cannot generate stealthy attack simulations in these situations.

In total, this process generated 326 successful lateral movement attacks, where each attack successfully accomplished its goal under the specified level of stealthiness. As seen in Table C.1, most of these simulated attacks fall under "non-stealthy" attack scenarios. For scenarios under the last two levels of stealthiness, we found that many users simply lacked any path to new credentials/machines that also exclusively moved between machines with prior logins. For example, in the targeted compromise scenarios, none of the starting victims had a path that both led to a set of elevated administrator credentials and also traversed a stealthy (previously trodden) path of logins to a sensitive machine; the typical paths to these critical servers often involve machines that only a handful of users (typically system administrators) legitimately access.

C.4 Additional Alert Details

Of the 2,399 alerts flagged as clear credential-switching paths (Section 6.6), 1,326 alerts reflect movement paths where the credential switching occurred in a login from a client to a server. These

	1	2	3	4	5	6	10	15	16	18	20
0.01	386	596	717	764	808	834	965	1,058	1,073	1,103	1,141
0.025	722	1,100	1,360	1,503	1,652	1,774	2,215	2,655	2,729	2,854	3,002
0.1	1,809	2,729	3,416	3,979	4,415	4,793	6,096	7,528	7,786	8,259	8,707
0.2	2,347	3,556	4,509	5,309	5,942	6,505	8,243	10,177	10,528	11,178	11,777
0.5	8,027	11,397	13,975	15,769	17,358	18,728	23,183	27,928	28,771	30,320	31,671
0.75	11,974	16,808	20,505	23,193	25,493	27,560	33,980	40,407	41,520	43,587	45,381

Table C.2: Baseline Comparison: the number of attacks that SAL [111] detects under different parameter combinations. Each column corresponds to a threshold value for an anomalous edge and each row corresponds to a minimum threshold value for the fraction of users or machines that a "benign login pattern" must match. For anomalous edge thresholds between 6–9, SAL identifies no additional attacks, so we omit these columns for space reasons.

alerts fall broadly into two categories of false positives. First, roughly 10% of these logins correspond to client machines that perform a login into a team-specific server with a rarely-used service account. These service usernames do not match the username of the client machine's owner, triggering an alert from Hopper's clear credential-switching detector; and because of the sparse usage of these service accounts, Hopper's procedure for inferring and pruning service account credentials does not filter out these alerts. The second class of false positive (approximately 70% of the false client-to-server alerts) results from system administrators in smaller offices performing a re-imaging or re-provisioning of an existing client machine. This machine re-provisioning process runs a script that generates many logins into management and authentication servers in order to properly configure the machine (e.g., downloading and install various software from internal repositories, configuring the machine policies and permissions with all of the organization's domain controllers, etc.). Each of these logins produces an alert for an unexpected use of system administrator credentials on a client device owned by another user (i.e., a path where Hopper believes the machine's old owner has suddenly started making login paths under a system administrator's credentials). Although our benign-scenario pruning (Section 6.6 and Appendix C.2) removes many clear credential-switching logins that result from machine (re)provisioning at Dropbox's major offices, a few of the smaller offices do not have dedicated subnets for machine provisioning. Consequently, the logins made as part of these machine provisioning operations do not get pruned by Hopper's filtering step and trigger alerts.

C.5 Baseline Evaluation Details

SAL consists of two stages. First, SAL takes a set of historical logins and uses this "training data" to construct a login graph analogous to Hopper's. Given a batch of new logins, SAL generates a set of candidate alerts (logins) by identifying all logins that traverse a graph edge that has occurred in $<$ N days in the training data, where N is a user-provided threshold that specifies the minimum number (or percent) of days for a "benign" edge. Second, SAL prunes these candidate alerts down to a final alert list by removing any login that matches a "benign login pattern". SAL uses its

	1	2	3	4	5	6	10	15	16	18	20
0.01	30	33	34	34	34	34	34	34	34	34	34
0.025	55	72	73	73	74	76	82	83	83	83	83
0.1	98	126	130	139	150	155	174	186	186	188	191
0.2	117	156	165	175	190	201	225	244	249	251	254
0.5	188	231	245	251	260	274	302	309	313	315	316
0.75	222	256	267	271	279	289	315	320	323	326	326

Table C.3: Baseline Comparison: the number of alerts generated by SAL [111] on our evaluation data under different parameter combinations.

historical training data to learn a set of benign patterns by mapping each login to a list of "login patterns", where each "login pattern" consists of a triplet of (source machine attributes, destination machine attributes, target user attributes). For example, given the login (src = Machine A, dest = Machine B, user = Alice), (source = New York, destination = San Francisco, user = Sales) would be one login pattern, if Machine A resides within New York, Machine B resides within San Francisco, and Alice works on the Sales team. SAL then aggregates all of the login patterns across its training logins and learns a set of "benign patterns" by identifying any login pattern where a sufficiently large fraction of source machines, destination machines, and/or users have at least one historical login that matches a pattern; e.g., given a user-specified threshold of 33%, SAL would add a login pattern to its benign pattern set if at least 33% of the enterprise's machines had a login that matched the pattern (as either the source or destination machine), or if over 33% of users had at least one login that matched the pattern.

Based on the data available to us, we use the following set of attributes from the SAL paper: each user has two attributes: (the user's team and the user's type: system administrator, regular user, or service account), and each machine has two attributes: (the machine's type: client or service and the machine's geographic location).

Because SAL requires two-user provided inputs, N the minimum number of days for a benign login edge and P the minimum fraction of machines or users for a benign pattern, we performed a grid-search over a range of parameters and selected the best parameters for our evaluation: i.e., the parameter combination that detected all attacks with the fewest false positives. Table C.3 and Table C.2 show the total alert volume and number of attacks detected under a range of these parameter combinations.