D^{2}M: Dynamic Defense and Modeling of Adversarial Movement in Networks

Scott Freitas [*] Andrew Wicker [†] Duen Horng (Polo) Chau [‡] Joshua Neil [§]

Abstract

Given a large enterprise network of devices and their authentication history (e.g., device logons), how can we quantify network vulnerability to lateral attack and identify at-risk devices? We systematically address these problems through D^2M, the first framework that models lateral attacks on enterprise networks using multiple attack strategies developed with researchers, engineers, and threat hunters in the Microsoft Defender Advanced Threat Protection group. These strategies integrate real-world adversarial actions (e.g., privilege escalation) to generate attack paths: a series of compromised machines. Leveraging these attack paths and a novel Monte-Carlo method, we formulate network vulnerability as a probabilistic function of the network topology, distribution of access credentials and initial penetration point. To identify machines at risk to lateral attack, we propose a suite of five fast graph mining techniques, including a novel technique called AnomalyShield inspired by node immunization research. Using three real-world authentication graphs from Microsoft and Los Alamos National Laboratory (up to 223,399 authentications), we report the first experimental results on network vulnerability to lateral attack, demonstrating D^2M's unique potential to empower IT admins to develop robust user access credential policies.

1 Introduction

Attack campaigns from criminal organizations and nation state actors are quickly becoming one of the most powerful forms of disruption. In 2016 alone, malicious cyber activity cost the U.S. economy between \$57 and \$109 billion [20]. These cyber-attacks are often highly sophisticated, targeting governments and large-scale enterprises to interrupt critical services and steal intellectual property [5]. Unfortunately, once an attacker has compromised a single credential for an enterprise machine, the **whole network becomes vulnerable to lateral attack movements** [8], allowing the adversary to eventually gain control of the network (i.e., escalating privileges via credential stealing [6]).

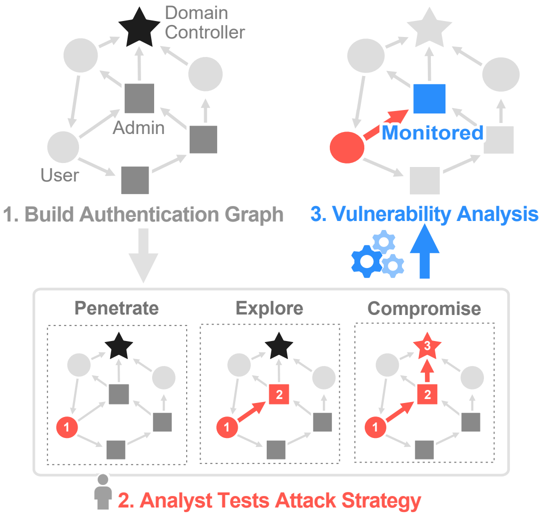
Domain Controller
Admin
User
1. Build Authentication Graph
Monitored
3. Vulnerability Analysis
Penetrate
Explore
Compromise
1
2
3
2. Analyst Tests Attack Strategy

Figure 1: Our D^2M framework: **1.** Builds an authentication graph from device authentication history; **2.** Allows security analysts to test different attack strategies to study network vulnerability; **3.** Identifies at-risk machines to monitor, preempting lateral attacks.

Despite their prevalence, observing and analyzing lateral attacks is challenging for multiple reasons: (1) lateral attacks are still relatively sparse compared to the unsuccessful attack; (2) attack ground-truth is hard to ascertain, and generally partially uncovered through investigation; (3) incident reports are frequently withheld from the public for security and privacy concerns; and (4) due to the fact that the adversary already has a valid credential for the network (e.g., gained through phishing [3]), attackers can operate as a legitimate user. While real attack data does exist—due to the above challenges, it is rarely fully visible, or accessible, making the study of a "complete" attack highly problematic.

Our Contributions

We propose D^2M, the first framework that systematically quantifies network vulnerability to lateral attack and identifies at-risk devices (Fig. 1).

Our major contributions include:

- **Attack Strategies** D^2M enables security researchers to integrate their crucial domain knowledge from studying prior attacks in the form of attack strategies. We developed three attack strategies by actively engaging researchers, engineers and threat

[*] Georgia Tech, Atlanta, GA (safreita@gatech.edu)

[†] Research while at Microsoft. Now at Uber, Seattle, WA (andrew.wicker@uber.com)

[‡] Georgia Tech, Atlanta, GA (polo@gatech.edu)

[§] Microsoft Corp, Seattle, WA (joshua.neil@microsoft.com)

hunters in the Microsoft Advanced Threat Protection group, whose expertise lies in tracking down adversaries in a post-breach environment (once adversary is on network). D^2M integrates real-world adversarial actions (e.g., privilege escalation), generating attack paths consisting of a series of compromised machines (Sec. 5; Fig. 1.2).

- **Network Vulnerability Analysis** We formulate a novel Monte-Carlo method for lateral attack vulnerability as a probabilistic function of the network topology, distribution of access credentials and initial penetration point (Fig. 1.3). This empowers IT admins to develop robust user access credential policies and enables security researchers to study the vulnerability of a network to lateral attack (Sec. 6).
- **Network Defense by Identifying At-risk Machines** To identify machines at risk to lateral attack, we propose a suite of five fast graph mining techniques, including a novel technique called AnomalyShield which prioritizes machines with anomalous neighbors and high eigencentrality (Fig. 1.3; Sec. 7).
- **Evaluation Using Real-World Data** Using three real-world authentication graphs from Microsoft and Los Alamos National Laboratory (LANL; up to 223,399 authentications), we report the first experimental results on network vulnerability to lateral attack and at-risk machine identification (Sec. 5).
- **Impact to Microsoft and Beyond.** The Microsoft Defender Advanced Threat Protection product is deployed to thousands of enterprises around the world, and is a leader in the Endpoint Detection and Response (EDR) market [13]. The ability to detect and prevent lateral movement is one of the most challenging areas of post-breach detection. This research has led to major impact to Microsoft products, inspiring changes to the product's approach to lateral movement detection.

Table 1 describes the main symbols used in the paper. We follow standard notation and use capital bold letters for matrices (e.g., $\boldsymbol{A}$), lower-case bold letters for vectors (e.g., $\boldsymbol{a}$) and calligraphic font for sets (e.g., $\mathcal{S}$).

2 Background and Our Differences

Our work intersects the domains of lateral attack and graph mining, we briefly review related work below. Different from existing work that detects lateral movement after an adversary is on the network, our work **quantifies network vulnerability to lateral attack** and **identifies at-risk machines**. Another important distinction is that this work uses real-world enterprise authentication graphs, while most prior work has not.

Symbol	Definition
G	Directed, unweighted, attributed graph
$\mathcal{V}$, $\mathcal{E}$	Set of nodes and edges in graph G
n, m	Number nodes $\|V\|$, edges in $\|\mathcal{E}\|$ in G
$\boldsymbol{A}(i,j)$	Adj. matrix of G at ith row, jth column
$\boldsymbol{u}(i)$	Eigenvector at position i
$\mathcal{C}, c$	Credential set; credential instance
D	Credential generation process
$\boldsymbol{d}$	Credential vector
$\mathcal{H}$, h	Ordered hygiene set; hygiene instance
$N^+(v)$, $N(v)$	Successors of v; neighbors of v
$\mathcal{R}$, $\mathcal{T}$	Set of start nodes; set of attacker moves
$\mathcal{S}_k$	Set of k nodes to monitor
$SV(\mathcal{S}_k)$,	Shield value of $\mathcal{S}_k$
$AV(\mathcal{S}_k)$	Anomaly value of $\mathcal{S}_k$
$L(G)$	Vulnerability of G to lateral attacks
$\boldsymbol{p}$	Attack path
$\boldsymbol{a}$	Per-machine anomaly vector
i_s	Number of sub-path intervals
k	Number of machines to vaccinate

Table 1: Symbols and Definition

2.1 Detecting Lateral Attacks Significant research in *detecting* lateral movement in networks has been done [14, 16, 19, 7]. Latte [14], a graph based detection framework, discovers potential lateral movement in a network using forensic analysis of known infected computers. In [16], Neil et al. detects lateral attacks using statistical detection of anomalous graph patterns (e.g., paths, stars) over time. Alternatively, Noureddine et al. [19] proposes a zero-sum game to identify which machines a defender should monitor to slow down an attacker. Finally, a data fusion technique is proposed by Fawaz et al. [7], where host-level process communication graphs are aggregated into system-wide communication graphs to detect lateral movement.

2.2 Graph Mining & Network Security Graph mining has been extensively applied to the more general domain of network security. Authentication graphs have been used to study network security from a variety of viewpoints [8, 11, 16]. In [8], Hagberg et al. studies credential hopping in authentication graphs and finds that by reducing a machine's credential cache, lateral movement can be restricted. Alternatively, Kent et al. [11] develops individual user authentication graphs to differentiate normal authentication activity from malicious. Orthogonal to the authentication graph and *our work*, attack graphs have been proposed to analyze a network's risk to known security issues [24, 2, 9]. These graphs represent sequences of known system vulnerabilities that can be maliciously exploited; and are often used by IT admins to determine patch priority.

3 Authentication Graph

D^2M converts authentication history of network devices into an *authentication graph*, where directed edges represent machine-machine authentications (i.e., logons) in an organization. Below, we provide an overview of the authentication graph setup and the infusion of real-world domain knowledge into its construction.

3.1 Building Graph Structure Modern enterprise computer networks typically rely on one of two types of centrally managed authentication mechanisms to authenticate user activity: Microsoft NTLM [1] or MIT Kerberos [17]. To avoid repeated authentication with network resources (e.g., printer, corporate web sites, email), both NTLM and Kerberos implement credential caching where user credentials are stored on the computer until either the user logs off (Kerberos), or the machine is restarted (NTLM) [8]. While these cached credentials are convenient for legitimate user activity, they pose significant risk for malicious exploitation [6, 25].

Leveraging this authentication history, we form a directed, unweighted graph $G = (\mathcal{V}, \mathcal{E})$, where an edge represents an authentication between source machine v_s and destination machine v_d (see Fig. 1.1). We combine all authentications between two machines into a single edge. These authentication events are recorded over a period of time, forming the graph topology of an organization [8, 11]. To verify that a remote connection between two machines can be established, authentication information is passed using cached credentials. In an enterprise network, these credentials typically follow a hierarchical scheme: *user* (c_1) at the bottom, *local admin* (c_2) and *network admin* in the middle (c_3), and *domain admin* (c_4) at the top ($c_1 < c_2 < c_3 < c_4$) [25]. Depending on the type of cached credential, it will be valid until the user logs out (Kerberos) or until the machine is restarted (NTLM).

3.2 Integrating Domain Knowledge To enhance D^2M with realistic security and attack practices, we integrate the following three components into our framework: (1) per-machine **credential caching**; (2) **network hygiene** (i.e., how many 'users' and 'admins' on the network); and (3) **domain controller** modeling.

Credential Caching We embed attribute information into graph G by giving each machine $v \in V$ a cached credential. These credentials are stored as a vector $\boldsymbol{d} \in \mathbb{R}^n$, where each entry is a machine in the authentication graph containing the most recent credential $\boldsymbol{d}(i) = c$. While some credential schemes have additional levels and queue lengths as active directory policies, our approach captures representative security information.

Network Hygiene We model various credential distributions through three levels of hygiene $h \in \mathcal{H}$ due to the unavailability of credential information in the network $\boldsymbol{d} =< c_1, c_2, ..., c_n >$ where $n = |\mathcal{V}|$. Each hygiene level (h_1: low, h_2: medium, h_3: high) represents the frequency with which credential types are observed on the network. Intuitively, a low hygiene level (h_1) models a network with loose IT policies and an abundance of high-level administrator credentials. In contrast, a high hygiene level (h_3) represents a network with strict IT policies and limited distribution of admin credentials. We select each hygiene distribution $h \in \mathcal{H}$ as: $h_1 = \{c_1\text{: } n,\ c_2\text{: } n/2,\ c_3\text{: } n/5,\ c_4\text{: } n/20\}$, $h_2 = \{c_1\text{: } n,\ c_2\text{: } n/4,\ c_3\text{: } n/10,\ c_4\text{: } n/50\}$ and $h_3 = \{c_1\text{: } n,\ c_2\text{: } n/8,\ c_3\text{: } n/20,\ c_4\text{: } n/80\}$, which are determined experimentally in conjunction with domain experts.

In practice, we distribute these credentials for a given hygiene h as follows. For every machine in the network $v \in \mathcal{V}$ we assign the lowest authorization level $\boldsymbol{d}(v) = c_1$. We then distribute higher level credentials as follows—for each increasing credential level $c \in \{c_2, c_3, c_4\}$, we randomly select $h(c)$ machines from $\mathcal{V}$ and loop through each one, replacing it's credential level with a higher one. While these distributions cannot match every organization's IT policies, we select them to model a broad range.

Domain Controller & Privilege Escalation The final component we model is the domain controller, which controls access to network resources. When a source machine v_s attempts to establish a remote connection to a destination machine v_d, the domain controller determines if v_s has sufficient privileges $\boldsymbol{d}(v_s) \geq \boldsymbol{d}(v_d)$. Since an organization's domain controller(s) are never observed with certainty, we identify it using PageRank (α=0.15) [21]—assigning the machine with largest PageRank vector $\boldsymbol{r} \in \mathbb{R}^n$ component the role of domain controller $v_{dc} = argmax(\boldsymbol{r})$. After discussions with domain experts, we make the simplifying assumption that the machine with largest PageRank is the domain controller v_{dc}, since it often has the largest number of incoming edges (from incoming authentication requests).

Finally, we incorporate the concept of privilege escalation by allowing the attacker to connect to a machine that is one credential level higher. That is, if the attacker has collected credentials c_1 and c_2, they can connect to a c_1, c_2, or c_3 machine. In practice, this is done through mining the memory of the machine to gain higher levels of credential [15].

4 Formulating the Research Problems

We formally define the three problems that D^2M addresses below. Then we present our solutions for them in **Section 5**, **6**, and **7** respectively.

Problem 1. ***Lateral Attack Modeling***

Given: *an attack strategy, an initial penetration point, and directed unweighted graph G with associated credential distribution $\boldsymbol{d} \in D$*

Find: *an attack path $\boldsymbol{p} =< v_1, v_2, ...v_i..., v_t >$ in graph G that starts from the penetration point and reaches the domain controller, while escalating privileges in increasing order (see Fig. 2)*

Problem 2. ***Lateral Attack Vulnerability***

Given: *graph G with credential distribution $\boldsymbol{d} \in D$*

Measure: *vulnerability $L(G)$ to lateral attacks*

Problem 3. ***Lateral Attack Defense***

Given: *graph G with credential distribution $\boldsymbol{d} \in D$, and suspected adversary movement $\boldsymbol{p}$*

Identify: *k best machines to monitor for attacks*

5 D^{2}M: Lateral Attack Modeling

We present our solution for the *lateral attack modeling* problem (Sect. 4: Problem 1). We begin with an overview of the lateral attack process in Section 5.1. Section 5.2 presents lateral attack strategies—developed with Microsoft domain experts—that produce lateral movement. Section 5.3 details the algorithm for modeling lateral attacks on authentication graphs.

5.1 Lateral Attack Overview An enterprise attack typically follows a kill chain, which can be distilled into three phases—(1) penetration of the network; (2) exploration of the network and escalation of privileges; and (3) exfiltration of data back to the command and control server [23]. We discuss each phase below and highlight our modeling assumptions.

Penetration An enterprise network is typically penetrated through two mechanisms—(a) phishing campaigns targeting organization employees or (b) incidental exposure from employees downloading malware on high-risk websites (drive-by download) [12]. We assume the former, since sophisticated adversaries often target enterprise networks for penetration. A phishing campaign begins by targeting organization employees through authentic looking emails containing malicious attachments or web links. These malicious attachments contain malware that installs a backdoor; once a backdoor is installed the attacker gains remote access to the machine, penetrating the enterprise network. We model this penetration process by assuming that most compromised employees (machines) $v \in \mathcal{V}$ are at the c_1 credential level and let the attacker randomly start on any of these machines $\mathcal{R} = \{v \in \mathcal{V} \mid \boldsymbol{d}(v) = c_1\}$.

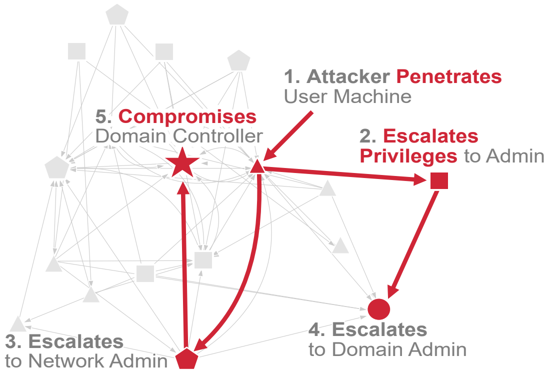

Figure 2: Attack path generated by D^2M. **1.** Network is penetrated; **2-4.** Attacker explores the network and escalates privileges; **5.** Attacker compromises the domain controller, gaining control of the network

Explore & Exploit Once an adversary is on a network, their goal is to explore the network and escalate privileges. This process begins by stealing the infected machines cached credentials, allowing them to authenticate with neighboring machines. These credentials can be stolen in a number of ways, however, it is beyond the scope of this work and we refer the reader to [25]. Once the adversary has connected to a neighboring machine, they again steal the cached credentials [6] and continue this process until they have obtained domain admin privileges c_4. We model this attack process in two ways—(1) black-box, where the attacker has no prior information on the network (i.e., normal pattern of authentications); and (2) gray-box, where the attacker has prior information on the network layout, possibly through prior reconnaissance or inside help.

Exfiltrate After the adversary has obtained a domain admin credential c_4, they're able to connect to any networked machine, freely exploring the network until they reach the domain controller. Upon accessing the domain controller, the attacker gains full control over the network. At this point the adversary can sweep the network for valuable information and exfiltrate with impunity. We leave modeling this aspect of the kill chain to future work.

5.2 Lateral Attack Strategies In conjunction with domain experts, we develop three attack strategies to model lateral attacks on authentication graphs; one black-box and two gray-box.

5.2.1 Black-Box Attack In the black-box setting we assume the attacker has no knowledge about the network and model movement through a modified random walk called *RandomWalk-Explore* (RWE).

RandomWalk-Explore (RWE) with 0.85 probability draws a machine v uniformly at random from the set of unvisited neighboring machines $\mathcal{T}$. With probability 0.15, the attacker randomly jumps with uniform probability to a machine in $\mathcal{R}$; this helps to model some of the usual behavior that can occur during an attack (e.g., when an attacker finds remote machine information in plain-text). In addition, we select 0.15 as the random jump probability to align with information retrieval literature [21]. We model the RWE process in Equation 5.1, which describes the probability mass function (PMF) of a discrete random variable X_1, which can take on any value in the range $R_{X_1} = \mathcal{T} \cup \mathcal{R}$ with probability $P_{X_1}(v)$.

$$P_{X_1}(v) = \begin{cases} 0.15/|\mathcal{R}|, & \text{if } v \in \mathcal{R} \\ 0.85/|\mathcal{T}|, & \text{if } v \in \mathcal{T} \\ 0, & \text{otherwise} \end{cases} \tag{5.1}$$

5.2.2 Gray-Box Attacks In the gray-box setting, the attacker has additional information in the form of the network topology—allowing for informed attack strategies. We propose two strateiges, *Rank-Explore* (RE) and *Degree-Explore* (DE).

Rank-Explore (RE) with 0.85 probability draws a machine v at random from the set of unvisited neighboring machines $\mathcal{T}$ with weight proportional to its PageRank vector r. With probability 0.15, the attacker randomly jumps with uniform probability to a machine in $\mathcal{R}$. This process is modeled in Equation 5.2.

$$P_{X_2}(v) = \begin{cases} 0.15/|\mathcal{R}|, & \text{if } v \in \mathcal{R} \\ 0.85 \cdot \boldsymbol{r}(v) / \sum\limits_{i \in \mathcal{T}} \boldsymbol{r}(i), & \text{if } v \in \mathcal{T} \\ 0, & \text{otherwise} \end{cases} \tag{5.2}$$

Degree-Explore (DE) with 0.85 probability draws a machine $v \in \mathcal{T}$ with weight proportional to the distribution of the network's degree vector $\boldsymbol{\delta} = diag(\boldsymbol{A} \cdot \boldsymbol{e})$. With probability 0.15, the attacker randomly jumps with uniform probability to a machine in $\mathcal{R}$. This process is modeled in Equation 5.3.

$$P_{X_3}(v) = \begin{cases} 0.15/|\mathcal{R}|, & \text{if } v \in \mathcal{R} \\ 0.85 \cdot \boldsymbol{\delta}(v) / \sum\limits_{i \in \mathcal{T}} \boldsymbol{\delta}(i), & \text{if } v \in \mathcal{T} \\ 0, & \text{otherwise} \end{cases} \tag{5.3}$$

Algorithm 1: Lateral Attack Modeling

```
Input: Adj. matrix $\boldsymbol{A}$, $h$, attack strategy
Result: Attack pattern $\boldsymbol{p}$
let $\boldsymbol{r}_o$ = PageRank($\boldsymbol{A}$); and $\boldsymbol{\delta}_o$ = diag($\boldsymbol{A} \cdot \mathbf{1}$)
let $\boldsymbol{d} \sim D_h$                    // distribute credentials
$\mathcal{R} = \{v \in V \mid \boldsymbol{d}(v) = c_1\}$                    // start nodes
$v = rand(\mathcal{R})$; let $\mathcal{T} = N^+(v)$; $p = [v]$
tried = {}; visited = {}
while $v \neq v_{dt}$ and $|\mathcal{T}| > 0$ and $|tried| < |\mathcal{T}|$ do
    $\mathcal{T} = \mathcal{T}$ / tried
    if $attack\_strategy == RWE$ then
        $v \leftarrow X_1$
    else if $attack\_strategy == RE$ then
        $\boldsymbol{r} = \boldsymbol{r}_o(\mathcal{T})$; $v \leftarrow X_2$
    else if $attack\_strategy == DE$ then
        $\boldsymbol{r} = \boldsymbol{\delta}_o(\mathcal{T})$; $v \leftarrow X_3$
    $\mathcal{T} = \mathcal{T} \cup$ tried
    if $Valid(v)$ and $v \notin visited$ then
        tried = {}
        $\mathcal{T} = \mathcal{T} \setminus \{v\} \cup N^+(v)$
        $\boldsymbol{p}$ += $v$; visited += $v$
    else
        tried $\cup$ $v$
Return $\boldsymbol{p}$
```

After a neighbor v has been selected by the attack strategy, we check that the attacker has the required credential level to visit this machine. For example, if c_2 is the current highest collected credential, then the attacker can move to any machine with credential level c_1, c_2, or c_3. If the move is valid, we update the set of unvisited neighbors $\mathcal{T}$ according to Equation 5.4 and allow the attacker to collect that machine's credential.

$$\mathcal{T} = \mathcal{T} \setminus \{v\} \cup N^+(v) \tag{5.4}$$

5.3 Lateral Attack Algorithm We allow the attacker to randomly penetrate various points of the network ($v \in R$) and then move according to one of the three strategies: RWE, RE and DE, until the domain controller v_{dc} is reached or there are no neighbors to visit. Each successful run of this simulation generates an attack path $\boldsymbol{p} =< v_1, v_2, ...v_i..., v_{dc} >$, representing the sequence of machines visited, with the last node v_{dc} representing the domain controller. This process is modeled in Algorithm 1 and repeated for multiple credential distributions $\boldsymbol{d} \in D$ to eliminate bias from a single distribution. An example attack path generated from Algorithm 1 can be seen in Figure 2.

5.4 Analysis of Lateral Attack Algorithm The time and space complexity of Algorithm 1 is $O(n^2)$ and $O(n + m)$, respectively.

There are two time expensive computations, PageRank $O(n)$; and attack strategy machine selection inside the while loop $O(n)$. Since the while loop can visit every node in the graph, the worst case complexity will be $O(n^2)$. Space is linear with respect to nodes and edges $O(n + m)$ in the graph. Detailed proofs are omitted to save space.

6 D^{2}M: Lateral Attack Vulnerability

We present our solution for the *lateral attack vulnerability* problem (Sect. 4: Problem 2). We begin by discussing the importance of network vulnerability scoring. We then formally introduce our method of measuring a network's vulnerability to lateral attacks. Finally, we discuss alternative graph vulnerability scores and why they are less suited to the task of measuring vulnerability to lateral movement.

Vulnerability Scoring To make data driven decisions regarding IT policy in an enterprise network, it is important to quantify the risk a network faces to lateral movement. Unfortunately, directly measuring this risk is difficult, requiring complex interactions of many unknown variables. To simplify these interactions, we propose to quantify network vulnerability to lateral attack $L(\cdot)$ as a function of three random variables—(1) network topology G, (2) distribution of credentials $\boldsymbol{d} \in D$ and (3) initial point of penetration $v \in \mathcal{R}$.

Since the true credential distribution $\boldsymbol{d} =< c_1, c_2, c_3, c_4 >$ is unknown, along with knowledge of the organizations IT policies (strict, loose: Section 3.1), we model credential distributions through the use of hygiene levels $h \in \mathcal{H}$. For a given hygiene level $h \in \mathcal{H}$, we can marginalize out the dependency of the vulnerability score to the credential distribution $\boldsymbol{d} \in D_h$ in expectation, reducing the vulnerability score to $L(G, \mathcal{H} = h, V = v)$. In addition, we can simulate the attacker penetrating many different points in the network $v \in \mathcal{R}$, allowing us to marginalize out the dependency to v and reduce the score to $L(G, h)$. We can view this process in Equation 6.5 through the lens of Monte Carlo simulation, where in expectation we compute the graph vulnerability across many different credential distributions $\boldsymbol{d} \in D$ and start nodes $v \in \mathcal{R}$.

$$L(G,h) = \frac{1}{|D_h} \frac{1}{|\mathcal{R}|} \sum_{\boldsymbol{d} \in D_h} \sum_{v \in \mathcal{R}} f(G, \boldsymbol{d}, v) \tag{6.5}$$

The vulnerability score $L(G, h)$ is a real number between $0 \leq L(G, h) \leq 1$, where a higher value indicates a more vulnerable network for the given topology G and hygiene level h. Intuitively, this score is saying that a network is more vulnerable if attacks are on average more successful for many credential distributions $\boldsymbol{d} \in D$ and penetration points $v \in \mathcal{R}$. We measure an attack's success through $f(\cdot)$, which simulates an attack using Algorithm 1. A value of $f(G, \boldsymbol{d}, v) = 1$ indicates a successful attack, which we define as being able to reach the domain controller v_{dc}. Future work could generalize this to other targets such as high value servers.

We further simplify the vulnerability score $L(\cdot)$ by marginalizing out the dependency to hygiene level $h \in \mathcal{H}$. This simplifies Equation 6.5 to a function of the network topology G, as seen in Equation 6.6.

$$L(G) = \sum_{h_i \in \mathcal{H}} p(h_i) \cdot L(G, h_i) \tag{6.6}$$

With no prior knowledge on the true distribution of hygiene levels in an organization, we assume a uniform prior $p(h) = 1/3$.

Alternative Scoring Significant work has gone into measuring the vulnerability of graphs [4, 22, 26]. For example, in [26] the authors define vulnerability of an undirected graph G as the largest eigenvalue $L(G) \triangleq \lambda$ of the adjacency matrix. The intuition is that as the largest eigenvalue increases, so does the path capacity of the graph. However, this form of topological vulnerability scoring can only indirectly measure the vulnerability of the graph to lateral movement since no security domain knowledge is integrated.

7 D^{2}M: Lateral Attack Defense

We present our solution for the *lateral attack defense* problem (Sec. 4: Problem 3), where the objective is to identify the best set of k machines $\mathcal{S}_k$ to monitor for lateral attacks. Once this set of machines $\mathcal{S}_k$ has been identified, multiple safeguards can be implemented, including: changing the sensitivity of on device machine learning models and force resetting the password.

We make the following assumptions during the defense process—(a) there exists per-machine anomaly detection models that alert on unusual behavior (e.g., deviation in port or process activity). Since behavioral deviations have a larger false positive rate, their behavior is anomalous but not necessarily malicious. For this reason, anomaly alerts are ill-suited for investigation in isolation due to low confidence. However, these deviation scores are useful for machine monitoring decisions, especially when these alerts aggregate together [16]. (b) We assume that each anomaly detection model is providing real-time feedback to the defender; and (c) that the defender views all anomalous activity as it occurs through the system alerts. While assumption (c) is strong, we leave it to future work to model partial information defense strategies.

7.1 Defense Strategies We propose a suite of five defense strategies, three *static* and two *dynamic*. A *static* strategy takes into account only the network topology G; useful for protecting machines when monitoring resources are limited. A *dynamic* strategy considers both the network topology G and suspected lateral path movement activity $\boldsymbol{p}^t, \boldsymbol{p}^{t-1}, ...\boldsymbol{p}^i..., \boldsymbol{p}^0$, where $p^i \in \mathbb{R}^n$ is a sub-path containing suspicious activity in a given interval. This could be useful for real-time protection malicious activity investigation.

Each attack path $\boldsymbol{p}$ is divided into i_s sub-paths, where each sub-path $\boldsymbol{p}^i$ is of equal size (except for, possibly, the last sub-path p^t) where $t \in [0, \lceil \frac{\boldsymbol{p}}{i_s} \rceil]$. A larger value of i_s creates a few long sub-paths, which could represent fast moving attacks in the network; conversely, a small i_s creates many short sub-paths, representing slow attacks.

Rank-Defense (RD) statically identifies at-risk machines based on the network's PageRank [21]. Assuming a sorted PageRank vector, we identify machines as follows: $\mathcal{S}_k = \cup_{i=1}^{k} \boldsymbol{r}_i$.

Degree-Defense (DD) statically vaccinates the network according to the machines in the network with highest degree. With a sorted degree vector, we identify machines as follows: $\mathcal{S}_k = \cup_{i=1}^{k} \boldsymbol{\delta}_i$. While RD and DD are simple defensive strategies, we are not aware of any work proposing to identify at-risk machines to lateral attacks using them.

NetShield (NS) [26] statically vaccinates the network according to the machine's Shield-Value (SV) in Equation 7.7. The actual selection of $\mathcal{S}_k$ occurs in conjunction with the NetShield algorithm from [26], where the intuition is to select nodes with highest eigencentrality [18] while enforcing distance between selected machines (small or zero $\boldsymbol{A}(i,j)$). Here, $\boldsymbol{A} \in \{0,1\}^{n \times n}$, λ is the largest eigenvalue, and $\boldsymbol{u}$ is the associated eigenvector.

$$SV(\mathcal{S}_k) = \sum_{i \in \mathcal{S}_k} 2\lambda \cdot \boldsymbol{u}(i)^2 - \sum_{i,j \in \mathcal{S}_k} \boldsymbol{A}(i,j)\boldsymbol{u}(i)\boldsymbol{u}(j) \tag{7.7}$$

Random Anomalous Neighbor Defense (RAND) dynamically identifies machines by selecting an anomalous machine v_a with weight proportional to its anomaly score $\boldsymbol{a}(v_a)$, where each element $\boldsymbol{a}(v) \in [0,1]$ and $\boldsymbol{a} \in \mathbb{R}^n$. We assume that when an alert is generated for a machine in a sub-path, it produces a value of $\boldsymbol{a}(v) = 1$, repeating for every machine $v \in \boldsymbol{p}^i$. After machine monitoring set $\mathcal{S}_k$ is identified using sub-paths $\boldsymbol{p}^i, ...\boldsymbol{p}^0$, the anomaly scores are decayed $\boldsymbol{a}^{t+1} = \boldsymbol{a}^t/2$ to give weight to recent activity (determined experimentally).

The RAND strategy in described through Equations 7.8 and 7.9. Eq. 7.8 describes the PMF of discrete random variable X_4, which can take on any value in the range $R_{X_4} = \{v \in \mathcal{V} \mid \boldsymbol{a}(v) > 0\}$ with probability $P_{X_4}(v)$. After drawing a machine $v_a \sim X_4$, we uniformly at random select a neighbor from v_a. This can be seen in Equation 7.9, which describes the PMF of discrete random variable X_5, where X_5 can take on any value in the range $R_{X_5} = N^+(v_a)$ with probability $P_{X_5}(v)$. This process repeats until k machines have been selected.

$$P_{X_4}(v) = \begin{cases} \boldsymbol{a}(v) / \sum\limits_{i \in \mathcal{V}} \boldsymbol{a}(i), & \text{if } v \in R_{X_4} \\ 0, & \text{otherwise} \end{cases} \tag{7.8}$$

$$P_{X_5}(v) = \begin{cases} 1/|N^+(v_a)|, & \text{if } v \in N^+(v_a) \\ 0, & \text{otherwise} \end{cases} \tag{7.9}$$

AnomalyShield (AS), a novel method we introduce for dynamic machine identification. We select machines for monitoring according to their ANOMALY-VALUE (AV) in Equation 7.10, in combination with ANOMALYSHIELD (Algorithm 2). The intuition is that we prioritize machines with anomalous neighbors and high eigencentrality.

$$AV(\mathcal{S}_k) = \sum_{i \in \mathcal{S}_k} \boldsymbol{u}(i) \sum_{j \in N(i)} \boldsymbol{a}(j)\boldsymbol{u}(j) \tag{7.10}$$

Since both NetShield and AnomalyShield use eigenvector centrality as the underlying centrality metric, we convert the directed authentication graphs to undirected ones for use in the strategies.

Algorithm 2: ANOMALYSHIELD

```
Input: Adjacency matrix $\boldsymbol{A}$, anomaly vector $\boldsymbol{a}$, and
       vaccination budget $k$
Result: a set $\mathcal{S}_k$ with $k$ nodes
Compute first eigenvalue $\lambda$ and corresponding
  eigenvector $\boldsymbol{u}$ of $\boldsymbol{A}$
$\boldsymbol{c}$ = $\boldsymbol{A}$ * (a * u)
score = $\boldsymbol{c}$ * $\boldsymbol{u}$
for $iter$ = 1 to $k$ do
    $v$ = argmax$_i$ score($i$), add $v$ to set $\mathcal{S}$
    score($v$) = -1
return $\mathcal{S}$
```

7.2 Analysis of Defense Strategies We evaluate time and space complexity with respect to each strategy since they are the dominating defense cost. The space is uniform across strategy $O(n + m + k)$, with time complexity shown below. Proofs omitted for space.

$$Time = \begin{cases} O(nlogn), & \text{if defense = RD} \\ O(nlogn), & \text{if defense = DD} \\ O(nk^2 + m), & \text{if defense = NS [4]} \\ O(kn + m), & \text{if defense = AS} \\ O(kn), & \text{if defense = RAND} \end{cases} \tag{7.11}$$

8 Experiments

8.1 Experimental Setup All experiments are conducted on three real authentication graphs, collected

Graph	Source	$\lvert V\rvert$	$\lvert E\rvert$	ρ	C	δ_{avg}
G_s	Microsoft	100	279	0.028	0.23	5.58
G_l	Microsoft	2,039	3,853	0.001	0.26	3.78
G_{lanl}	LANL	14,813	223,399	0.001	0.62	30.16

Table 2: Graph Statistics. ρ: graph density, C: average clustering coefficient, δ_{avg}: mean node out-degree.

over 30 days (statistics in Table 2). Two graphs are from Microsoft: anonymized enterprise networks G_s and G_l; and one is from Los Alamos National Lab [10]: open-sourced network G_{lanl}. For each attack strategy and hygiene level, we strive to collect 200 unique attack paths for 50 credential distributions $\boldsymbol{d} \in D$. These parameters are determined based on the available 2-week computation budget for data collection. Certain combinations of G and $\boldsymbol{d}$ have a high rate of attack failure; we terminate the collection process at 10,000 failed attempts, collecting as many as possible.

8.2 Network Vulnerability Analysis In Table 3, we summarize the first experimental results on network vulnerability to lateral attack by analyzing the attack strategies *Rank-Explore* (RE), *Degree-Explore* (DE), and *RandomWalk-Explore* (RWE) (discussed in Sect. 5). For each strategy, we average the attack path length across all credential distributions. We compute the network vulnerability statistics using Eq. 6.5—hygiene-specific $L(G, h)$; and Eq. 6.6—whole-network $L(G)$ from Section 6. We identify multiple key insights:

1. **Informed Strategies Lead to Quicker Attacks** The RE and DE strategies produce shorter paths in general, compared to RWE. This is expected, as prior knowledge should help the attacker reach the domain controller in less time. Also, adversaries likely prefer shorter attack paths, which leaves smaller footprints for anomaly systems to detect.
2. **Improving Hygiene Reduces Vulnerability** Increasing network hygiene ($h_1 \rightarrow h_2 \rightarrow h_3$) causes longer attack paths (or none at all) and generally reduces vulnerability (e.g., for G_s and G_l). On graph G_s, the highest hygiene level h_3 critically reduces high-level admin credentials, significantly improving network robustness (vulnerability reduced to 0). Such findings can empower IT admins to develop robust user access credential policies.
3. **Linking Topology to Network Vulnerability** Networks that are well-connected are more vulnerable to lateral attack (e.g., G_{lanl}, with higher average clustering coefficient and node degree). This is expected, due to increased lateral movement opportunities. Relatedly, improving network hygiene level in such a well-connected network does not seem to reduce network vulnerability.

		Avg. Path length			**Vulnerability**	
Graph	**Hygiene**	**RE**	**DE**	**RAND**	$L(G,h)$	$L(G)$
G_s	h_1	19	19	25	.773	.525
	h_2	49	39	39	.801	
	h_3	0	0	0	0	
G_l	h_1	33	36	46	.005	.005
	h_2	63	63	68	.006	
	h_3	133	139	139	.004	
G_{lanl}	h_1	22	18	45	.967	.976
	h_2	88	128	90	.981	
	h_3	-	-	249	.981	

Table 3: Vulnerability Statistics. Statistics excluded for G_{lanl} strategies RE and DE in h_3 as computation exceeded budget (Sect. 8.1).

8.3 Defense Strategy Analysis We report the first results for identifying machines at-risk to lateral attack, evaluating each defense strategy proposed in Section 7. We measure the success of each strategy by its ability to predict attacker movement. That is, given graph topology G and suspected lateral attack movement $\boldsymbol{p}^i, ..., \boldsymbol{p}^0$, predict attack activity at $\boldsymbol{p}^{i+1}$ (each $\boldsymbol{p}^i$ is a sequence/path of suspected machines traversed by the attacker). Formally, we intersect the predicted *at-risk machines* $\mathcal{S}_k$ with $\boldsymbol{p}^{i+1}$. Since the defender likely monitors the domain controller, we exclude it from $\mathcal{S}_k$. We repeat this process for each sub-path (except $\boldsymbol{p}^0$) and average over all attack paths. Figure 3 shows every combination of attack and defense strategy, with budget k=8 and hygiene h_2, which provide representative results. We identify multiple key insights:

1. **AnomalyShield as Effective General Defense** AnomalyShield generally performs well (identified more machines) across: network topology (rows in figure), adversary's prior knowledge (columns), and attack speed (horizontal axes). We believe this is because AnomalyShield focuses on high-centrality machines with anomalous neighbors, combining desirable attributes from static and dynamic methods.
2. **Similar Effectiveness in Small Graphs** All strategies perform similarly in small graph G_s (first row), since fewer machines exist for monitoring.
3. **Large Graphs Require Informed Defense** Uninformed defense strategy RAND is significantly less effective in the large graph G_{lanl} (last row), especially when encountering faster attacks. This could be explained by the need for intelligent decision making in the presence of many options.

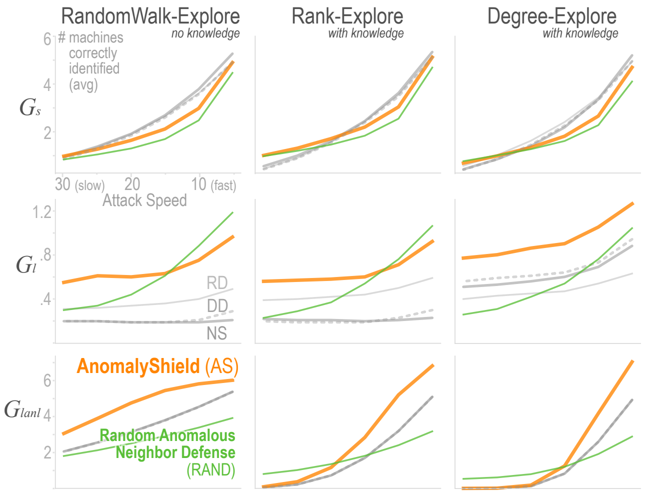

Figure 3: Each defense strategy is compared on three graphs and attack strategies, where AnomalyShield performs well across a majority of application scenarios.

9 Conclusion

We present D^2M, the first framework that systematically quantifies network vulnerability to lateral attacks and identifies at-risk devices. D^2M models lateral attacks on enterprise networks using attack strategies developed with Microsoft. We formulate network vulnerability as a novel Monte-Carlo method and propose a suite of five fast graph mining techniques, including the novel AnomalyShield method, to identify at-risk machines. Using real data, we demonstrate D^2M's unique potential to empower IT admins to develop robust user access credential policies.

10 Acknowledgements

This work was in part supported by the NSF grant IIS-1563816 and GRFP (DGE-1650044).

References

[1] *The ntlm authentication protocol and security support provider*, tech. report, 2006.

[2] P. Ammann, D. Wijesekera, and S. Kaushik, *Scalable, graph-based network vulnerability analysis*, in CCS, ACM, 2002, pp. 217–224.

[3] M. N. Banu and S. M. Banu, *A comprehensive study of phishing attacks*, IJCSIT, 4 (2013), pp. 783–786.

[4] C. Chen, H. Tong, B. A. Prakash, C. E. Tsourakakis, T. Eliassi-Rad, C. Faloutsos, and D. H. Chau, *Node immunization on large graphs: Theory and algorithms*, TKDE, 28 (2015), pp. 113–126.

[5] CrowdStrike, *Blurring the lines between statecraft and tradecraft*, Global Threat Report.

[6] S. Duckwall and C. Campbell, *Hello my name is microsoft and i have a credential problem*, Blackhat USA 2013 White Papers, (2013).

[7] A. Fawaz, A. Bohara, C. Cheh, and W. H. Sanders, *Lateral movement detection using distributed data fusion*, in SRDS, IEEE, 2016, pp. 21–30.

[8] A. Hagberg, N. Lemons, A. Kent, and J. Neil, *Connected components and credential hopping in authentication graphs*, in SITIS, IEEE, 2014, pp. 416–423.

[9] S. Jha, O. Sheyner, and J. Wing, *Two formal analyses of attack graphs*, in Proceedings 15th IEEE Computer Security Foundations Workshop. CSFW-15.

[10] A. D. Kent, *Cybersecurity Data Sources for Dynamic Network Research*, in Dynamic Networks in Cybersecurity, Imperial College Press, June 2015.

[11] A. D. Kent, L. M. Liebrock, and J. C. Neil, *Authentication graphs: Analyzing user behavior within an enterprise network*, Computers & Security, 48 (2015).

[12] V. L. Le, I. Welch, X. Gao, and P. Komisarczuk, *Anatomy of drive-by download attack*, in ACSW-AISC, Australian Computer Society, Inc., 2013, pp. 49–58.

[13] R. Lefferts, *Gartner names microsoft a leader in 2019 endpoint protection platforms magic quadrant.*

[14] 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 MILCOM, IEEE, 2018, pp. 1–6.

[15] J. Mulder, *Mimikatz overview, defenses and detection*, 2016.

[16] J. Neil, C. Hash, A. Brugh, M. Fisk, and C. B. Storlie, *Scan statistics for the online detection of locally anomalous subgraphs*, Technometrics, (2013).

[17] B. C. Neuman and T. Ts'o, *Kerberos: An authentication service for computer networks*, IEEE Communications magazine, 32 (1994), pp. 33–38.

[18] M. E. Newman, *Mathematics of networks*, The new Palgrave dictionary of economics, (2016), pp. 1–8.

[19] M. A. Noureddine, A. Fawaz, W. H. Sanders, and T. Başar, *A game-theoretic approach to respond to attacker lateral movement*, in GameSec, Springer, 2016.

[20] T. C. of Economic Advisers, *The cost of malicious cyber activity to the u.s. economy*, (2018).

[21] L. Page, S. Brin, R. Motwani, and T. Winograd, *The pagerank citation ranking: Bringing order to the web.*, tech. report, Stanford InfoLab, 1999.

[22] S. Saha, A. Adiga, B. A. Prakash, and A. Vullikanti, *Approximation algorithms for reducing the spectral radius to control epidemic spread*, in SDM'15.

[23] J. Sexton, C. Storlie, and J. Neil, *Attack chain detection*, Statistical Analysis and Data Mining: The ASA Data Science Journal, 8 (2015), pp. 353–363.

[24] O. Sheyner and J. Wing, *Tools for generating and analyzing attack graphs*, in International Symposium on Formal Methods for Components and Objects.

[25] M. Soria-Machado, D. Abolins, C. Boldea, and K. Socha, *Detecting lateral movements in windows infrastructure*, CERT-EU Security Whitepaper 17–002.

[26] H. Tong, B. A. Prakash, C. Tsourakakis, T. Eliassi-Rad, C. Faloutsos, and D. H. Chau, *On the vulnerability of large graphs*, in ICDM.