Towards Foundational Security Metrics

Nilofar Mansourzadeh
Carleton Internet Security Lab
Carleton University
Ottawa, ON, Canada
nilofarmansourzadeh@cmail.carleton.ca

Anil Somayaji
Carleton Internet Security Lab
Carleton University
Ottawa, Ontario, Canada
soma@scs.carleton.ca

Abstract

A foundational problem in computer security is determining to what extent a system is secure. While security metrics allow various aspects of security to be quantified, individual and even aggregated metrics cannot tell us whether a system is secure in an absolute or relative sense. We propose that metrics based on attacker knowledge reuse could provide a foundation for understanding the security of a system from a holistic perspective. In this paper we review the foundations of existing security metrics, propose two new metrics based on knowledge reuse, and discuss how knowledge reuse-based metrics could be used to inform theoretical and practical security analysis.

CCS Concepts

• **Security and privacy → Formal security models**.

Keywords

security metrics, knowledge reuse, security foundations

ACM Reference Format:
Nilofar Mansourzadeh and Anil Somayaji. 2024. Towards Foundational Security Metrics. In *New Security Paradigms Workshop (NSPW '24), September 16–19, 2024, Bedford, PA, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3703465.3703467

1 Introduction

What does it mean for a system to be secure? If a system has authentication and access control methods, protects data integrity and privacy with cryptography, and is kept up-to-date with the latest software patches, we may want to assume it is more secure than one without such protections, all else being equal. However, if we introduce protections such as physical access restrictions and limited network access, or if defense mechanisms are misconfigured in some way—say, by installing a default account with administrative access and no password—the relative security of systems change.

Note that this question is not directly related to technology; instead, it is an epistemological question: it is not about what we know, but how do we know what we know. In software engineering a focus on epistemology has helped improve the state of the art [12]. In computer security, researchers have criticized the lack of scientific rigor in computer security [3, 4, 24] and many have proposed and studied security metrics [13, 14] even while others have doubted the benefits of quantifying security [27]. In our opinion, however, the fundamental question of what does it mean for a system to be secure remains unanswered.

We assert that in the absence of perfectly secure computer systems, security is a relative concept that is inversely proportional to the knowledge of attackers. A given system configuration can be considered relatively secure today but hopelessly insecure tomorrow not because the system has changed, but because attackers have learned new attacks. Successful defense, then, is ultimately predicated on managing the impact of changes in the knowledge available to attackers. While "security through obscurity" is an aspect of such knowledge management, it is just one strategy defenders have at their disposal. Cryptographic key management, software updates, anti-malware signatures, firewalls—these and more all impact the ability of attackers to use the knowledge they obtain to compromise the security of their targets.

To measure the security of a system in a general sense, the above suggests that we need metrics that capture the dynamics of attacker knowledge—how attackers gain an advantage through innovation and how defenders in turn reduce that advantage. Knowledge is a much more abstract concept than information and itself is not so easily quantified [2]; however, just as performance metrics are often compared to idealized benchmarks, appropriate idealized security metrics could help guide the creation of more secure systems.

Our key contribution here are two security metrics, knowledge obfuscation and defense evolution, that help capture the knowledge dynamics between attackers and defenders. These metrics, on their own, can only be calculated in an abstract sense, given that they depend on factors that cannot be readily measured. As we explain, however, they can nevertheless be used to guide system design and to evaluate the relative utility of other security metrics.

The rest of this paper proceeds as follows. We discuss past work related to evaluating security and security metrics in Section 2. We then further discuss our motivation for this work in Section 3 and the basis of our approach in Section 4. Section 5 presents our model of attacker knowledge reuse, and Section 6 presents our two metrics based on knowledge reuse. Section 7 discusses how our metrics could be used to improve the security analysis of systems. We conclude with a discussion of the contributions, limitations, and implications of our work in Section 8.

2 Evaluating Security

Security evaluation is a complex topic that runs through the entire field of computer security. For example, when a vulnerability is found, we must determine its potential impact before determining how it should be addressed. Similarly, it is not enough to create

NSPW '24, September 16–19, 2024, Bedford, PA, USA

ACM ISBN 979-8-4007-1128-2/24/09
https://doi.org/10.1145/3703465.3703467

a defense; we must also be able to determine its ability to stop attacks, otherwise we will not know whether it is worthwhile for it to be deployed. Attackers also do security evaluations in order to determine how to potentially exploit a target. To try and understand the problem of security evaluation, here we discuss work from three perspectives: cryptography, attack modeling, and security metrics. While this classification is far from comprehensive, it can help us understand the potential benefits of more holistic approaches to evaluating security.

Cryptography has long been at the forefront of efforts to create precise definitions of security, as the security of all cryptographic primitives and protocols are grounded in threat models that encode specific assumptions about attacker capabilities. For virtually all cryptographic constructs, it is assumed that attackers have full knowledge of how the protocols or primitives work. Further, by necessity, it is also assumed that 1) defenders execute the specified operations precisely and 2) attackers have bounded capabilities in terms of their available analysis tools and computational capabilities. Small changes to many cryptographic constructs invalidate their security properties—One time pads, which are provably secure, get broken when defenders reuse keying material or when the keys are not perfectly random [18]. Unbounded capabilities invalidate entire categories of cryptographic defenses—DES became increasingly insecure as the cost to brute force a 56-bit key became sufficiently low [10].

From a security analysis perspective, the challenge of cryptography arises from the mismatch between theoretical cryptographic security guarantees and their strength in practical contexts. Keys are compromised through software vulnerabilities and improper key management; digital signatures are effectively invalidated through the corruption of certificate trust chains; secure protocols are compromised through side channels and attacks on the endpoints. Cryptographic threat models do not capture the means or difficulty of such non-cryptographic attacks; to better understand their impact on overall security, we need other approaches.

Many common approaches to modeling system security focus on how attackers identify individual exploits and leverage them to compromise systems or networks. For example, approaches such as attack graphs [22], stepping stones [30], and topological vulnerability analysis [7] have been used to assess the relative vulnerability of systems. While conceptualizing how an attacker could access a system allows threat models to be refined and defenses to be tightened, at a conceptual level they capture specific slices of the conflict between attackers and defenders, as it is essentially impossible to capture all possible paths in and through a system—there are always new opportunities. However, models such as attack graphs can also be used to model cybersecurity risk and attacker capabilities in more general ways, as evidenced by the TREsPASS project [15].

Game theory has been used by some researchers to model interactions in computer security [11, 19]. While game theory can represent specific attacker/defender interactions, the very nature of game theory limits its utility in computer security because game-theoretic games have well-defined rules and actions that each player can take. In contrast, in computer security attackers regularly invent new "moves," ones that can often break the "rules" that defenders thought they were playing by, forcing defenders to revise their assumptions and figure out how to respond.

Security metrics is a complex topic, with multiple surveys [14, 27, 29] and books [5, 8]; as such, a full review is not feasible here. However, a review of some key work and observations about patterns in the literature is sufficient to highlight the gaps we see in past work.

Security metrics quantify virtually any aspect of system configuration or history that could be related to security. Some measures are historical, such as how often vulnerabilities have been found, how long it took for vulnerabilities to be patched, and the time between patches being made available to being applied to a given system. Some are empirical, arising from lab or field studies where a defense system's ability to detect malware or intrusions are tested. Others are more theoretical, such as password strength or address space randomization entropy, where the construction of the system should give certain security properties but its real-world security is impacted by many external factors[1]. No matter the type of metrics, however, each is only capturing a small aspect of system security.

3 Motivation

As explained in the previous section, existing security metrics capture narrow slices of the security state of a system because they are focused on things that can be measured historically, empirically, or theoretically. Paradoxically, then, to develop metrics that give a broader view of security, we must focus on patterns that cannot be so easily measured but can still be conceptualized.

But then, what is the point of a metric that cannot be calculated in practice? As it turns out, such metrics are actually commonplace in other contexts. Goodness, quality, normality, convenience, attractiveness—virtually anything that people care about can be thought of as a metric, but one that exists in a abstract, conceptual space rather than in the world of things that can be measured explicitly. Such idealized metrics serve as a bridge between what can be measured and our actual goals.

Unlike these other abstract metrics, security is a much more difficult abstract metric to work with because security is dictated by the actions of adaptive adversaries. What is secure today is no longer secure tomorrow because of attacker innovation. Thus, even thinking about something being "more secure" or "less secure" is much less grounded than similarly abstract concepts such as quality or attractiveness. What we need, then, are abstract metrics that capture what we mean when we talk about security, but do so in a way that captures attacker/defender adaptation dynamics.

The purpose of this work, then, is to develop alternative abstract metrics that capture the dynamics between attackers and defenders. We propose that a more holistic view of computer security can be grounded in knowledge reuse formalized using ideas from artificial intelligence. We explain further in the next section.

4 Knowledge Reuse and AI

Attacks on computer systems succeed when an attacker has the right knowledge. This knowledge could be a password, a script exploiting a vulnerability, or the information and skills needed to carry out a social engineering attack. To stop or mitigate such attacks, defenders require their own knowledge: that a password has been compromised and should be changed, a software patch

[1] Cryptographic security measures are virtually all theoretical.

to eliminate a vulnerability, or training to make social engineering attacks harder. Attackers are thus incentivized to increase their knowledge, and defenders are incentivised to do what they can to invalidate attacker knowledge. To develop abstract metrics that capture the above dynamic, we need to formalize knowledge. Fortunately, the field of artificial intelligence has already developed formal methods for representing knowledge, particularly through search. We explore this idea below.

AI algorithms and techniques are complex computational processes designed to mimic human cognition, thereby enabling machines to solve problems, make decisions, and learn from experience [20]. These methodologies include machine learning, deep learning, natural language processing, and reinforcement learning [25]. Expert systems, decision trees, fuzzy logic, and genetic algorithms are also techniques used in AI [16].

In the context of AI, a search space is the domain or range of all possible configurations where the solution to a problem can be found. It is also known as the state space, particularly in problems modeled as state-space searches [20]. This includes the set of all states reachable from the initial state by any sequence of actions. Each "state" in the search space represents a different configuration of the problem. For instance, if you were using AI to solve a chess game, the search space would include all possible legal positions of pieces on the board. If you were using AI to find the shortest route between two cities, the search space would include all possible paths that could be taken. Exhaustive search algorithms such as depth-first search and breadth-first search can be used to explore large search spaces at great computational cost. Many AI algorithms, from A* search to training algorithms for deep neural networks, can be seen as alternative ways to search large search spaces, ones that work better or worse depending upon the nature of the space.

Knowledge creation in AI, particularly through the exploration of a search space, occurs as algorithms investigate different potential solutions or states and learn from the outcomes of these explorations. This process is extensively studied in reinforcement learning (RL), a branch of AI [25]. In RL, an agent interacts with an environment (which is essentially the search space) by taking actions, transitioning between states, and receiving rewards or penalties. The agent starts with little or no knowledge about the environment. As it explores the search space (i.e., tries different actions in different states), it begins to learn which actions lead to higher rewards in specific states, gradually creating a policy—a mapping of states to the best action in that state. This policy can be considered as the knowledge that the agent has acquired about the environment.

Similarly, in a supervised learning scenario such as training a neural network to classify images, the search space might be the set of all possible values of the network's weights and biases. The learning process involves searching this space to find the values that minimize the difference between the network's predictions and the true labels of the training data. The knowledge created here is the optimal or near-optimal set of weights and biases that the network can use to classify images accurately. In both scenarios, the search space's exploration is a critical part of the learning process, leading to the creation of new knowledge as the AI system discovers which configurations of its parameters or actions lead to the best outcomes.

In our research, we remain agnostic to the specific AI algorithms or manual search strategies that an attacker may employ. We acknowledge that much like in any other problem-solving scenario, an attacker navigates a domain of possibilities—a search space—to improve their understanding of it and to discover solutions. This exploration process leads to the creation of knowledge that can be used to invalidate previously held assumptions. Irrespective of the AI methods used, altering the search space changes the difficulty of finding a solution. While many other factors can influence it, the difficulty of a search is intrinsically related to the size of the search space.

This concept aligns with the well-established 'No Free Lunch Theorem' [28] in machine learning and optimization, which essentially posits that no single algorithm is superior when averaged across all possible problems. The theorem implies that the effectiveness of an AI algorithm depends on the specific characteristics of the problem at hand, including the size and structure of the search space. Thus, while an algorithm may excel in certain search spaces, its performance may diminish in others.

While work in AI can give us insights into the relationship between knowledge and security, we acknowledge that there are other perspectives on the nature of knowledge and these perspectives can also help us understand the problem of computer security. In particular, epistemology is a branch of philosophy that examines the nature of knowledge. From an epistemological perspective, knowledge is not simply information but is contextual and dynamic, shaped by both individual and collective experiences [26]. Such a social perspective on knowledge is not directly captured by the formalism we present in the next section, and indeed any formal model is unlikely to capture the nuances and complexities of epistemology in the context of security or any other field. Like any model, however, ours is distinguished as much by what it leaves out than by what it includes.

In the upcoming section, we delve into a formalization of exploring a domain of possibilities, commonly referred to as a search space. This formalization serves as the basis for our proposed metrics.

5 Modeling Attacker Knowledge

In this section, we present a model that shows the dynamics of computer security are best understood as a unique pattern of knowledge reuse, one in which attackers develop knowledge of how to compromise systems.

An attacker has to create knowledge in order to develop an exploit and attack a system, but how much knowledge has to be created and how we model the knowledge creation? In AI knowledge creation is modeled as search over an appropriate search space, with search leading to knowledge. Here, we are doing the same thing. We define **Attack** as search for any form of malicious action and **Attacker** as an agent who performs the search.

Attackers always build knowledge based on past knowledge like personal experience or the work of others. The more background they have, the easier knowledge creation is. Knowledge reuse becomes a restriction on the search space. Search spaces are used in security most commonly in cryptography. In this research, we are generalizing this to apply to security threats in general by using a more abstract search space.

5.1 Definitions

In this section, we begin by defining some terms that will be used in our model. We begin by understanding the foundational concept of an attack search space.

Imagine you have a house. The attack search space for a burglar would be all the possible entry points into your house—the front door, back door, windows, the chimney, etc. In the context of computer security, if your house is a computer system or network, the attack search space would include software vulnerabilities, weak passwords, open ports, and so on.

The attack search space describes all the possible ways an attacker could attempt to compromise a system. It is essentially the universe of all potential avenues an attacker might explore to achieve their malicious intent.

Definition 1 (**Attack Search Space**). *Let A be the attack search space. Each element $a \in A$ represents a possible attack vector. For instance, in our house analogy, a_1 might be the front door, a_2 the back door, and so on. The size of this search space is represented by $|A|$, which is the number of elements in the set A.*

$$A = \{a_1, a_2, ...\} \tag{1}$$

Note that A can be used to represent the search space of an entire system (a computer or an entire enterprise network) or it can represent attacks that target a specific subsystem (such as an organization's Active Directory).

To better visualize the concept of the attack search space, let us consider a few examples.

Example 1 (PINs). Consider a simple login form. If an attacker wants to brute-force the password, the attack search space consists of all possible password combinations. If the password is a 4-digit PIN, then $|A| = 10^4 = 10{,}000$ possible combinations.

Example 2 (Buffer Overflow Vulnerabilities). Consider a software application that takes user input without properly limiting its size. The attack search space for exploiting buffer overflow vulnerabilities in this application involves finding inputs that not only exceed the buffer's allocated space but also successfully execute arbitrary code. Unlike a finite set of PIN combinations, this space includes a vast range of inputs varying in length, content, and structure.

Example 3 (Authentication Verification Errors). Consider a web application with multiple endpoints requiring user authentication. The attack search space related to authentication verification errors encompasses all possible ways an attacker might bypass authentication or escalate privileges without valid credentials. This space is abstract and multifaceted, involving various methods such as session hijacking, forging authentication tokens, exploiting logic flaws, and more.

The attack search space is a crucial concept in computer security. By understanding the size and scope of this space, defenders can better prepare and protect systems, and attackers can determine the feasibility of their potential methods. The bigger the attack search space, the harder (and often longer) it generally is for an attacker to successfully compromise a system. However, attackers look for ways to reduce the effective size of this space by choosing specific types and avenues for attack, and by developing better techniques for searching for specific vulnerabilities.

Building on the concept of the attack search space, we now delve into the specific pathways or methods an attacker might employ, termed as attack vectors.

Definition 2 (**Attack Vector**). *An Attack Vector is a representation of the way or approach an attacker uses to exploit a system. It encapsulates three main elements that are crucial to understanding the nature of the attack.*

Given an attack vector a:

$$a = (M, T, E) \tag{2}$$

Where

- *M represents the method or technique used by the attacker. It is the how of the attack.*
- *T denotes the target of the attack. It is the where or what of the attack.*
- *E encapsulates the conditions that need to be in place for the attack to be successful. These could be vulnerabilities in the system, user behaviors that can be exploited, or even external conditions like a natural disaster that an attacker is taking advantage of.*

To further illustrate this idea, let us examine a real-world scenario involving an SQL Injection attack.

Example 4 (Attack Vector). In the example below, we are looking at one of the most common web attack vectors: the SQL Injection attack.

- M: The method being used by the attacker is SQL Injection. This technique involves injecting malicious SQL code into input fields to manipulate or query the database in unintended ways.
- T: The target of this attack is the Web application's user login page. This means the attacker is specifically trying to exploit the login page of a web application, possibly to gain unauthorized access.
- E: The exploit conditions here are that the application does not sanitize user input, and the database directly processes the raw input. This is a common vulnerability in web applications where user inputs are not checked or sanitized for malicious content before being processed. If an application does not sanitize inputs, it might execute malicious SQL code provided by the attacker.

For this particular attack vector, we can represent it as:

$a_{SQLinjection}$ = (SQL Injection, User login page, No input sanitation)

Now that we have explored the specifics of attack methods, it is crucial to consider the knowledge attackers might possess about vulnerabilities and how they might reuse this knowledge.

Definition 3 (**Attacker's Knowledge**). *The attacker's knowledge about specific vulnerabilities refers to the information they have about potential weaknesses in the system. Let K be the set of vulnerabilities the attacker knows. Each vulnerability $k \in K$ represents a specific weakness in the system.*

Definition 4 (**Knowledge Reuse**). *Reusing knowledge means leveraging insights or methods from past attacks for new attacks. Knowledge reuse can be formalized as a function R, which maps a known vulnerability to potential attack vectors in the search space.*

$$K = \{k_1, k_2, ...\} \tag{3}$$

$$R : K \rightarrow A \tag{4}$$

To contextualize the concept of knowledge reuse, let us envision a scenario where an attacker leverages known vulnerabilities used in past exploits.

Example 5 (Knowledge Reuse). Imagine a hacker who, in the past, exploited a vulnerability in System A. Now, they come across System B and recognize that it has the same vulnerability. Instead of starting from scratch, the hacker can reuse the knowledge obtained from their past exploitation of System A to quickly and efficiently attack System B.

This is analogous to a locksmith who knows how to pick a specific type of lock. If they encounter the same lock type on a different door, they can use their prior knowledge to open it without having to figure out the mechanism all over again.

The idea of an attacker's knowledge and knowledge reuse underscores the importance of regularly updating systems and fixing known vulnerabilities. If attackers can reuse their methods from past exploits, it makes their job easier and faster. On the defense side, understanding these concepts helps in predicting potential threats and deploying appropriate countermeasures. If a vulnerability is known and fixed in one system, it is essential to ensure that other similar systems are also patched to prevent knowledge reuse by attackers.

With the understanding of how knowledge reuse functions, we now investigate its direct implications on the nature of the attack search space.

Theorem 6. **(Knowledge reuse reduces effective attacker search space.)** *Let A denote the full attack search space and K represent the set of known vulnerabilities within this space. The application of knowledge reuse in identifying attack vectors reduces the search space from A to a smaller subset A', where $A' = R(K)$ and R maps known vulnerabilities to their respective attack vectors. This reduced search space A' has a size that is less than or equal to the size of A, formally expressed as $|A'| \leq |A|$. Moreover, the subset A', being informed by prior knowledge (K), possesses a higher density of exploitable vulnerabilities compared to the remaining portion of A not covered by K.*

Proof. Assume the full attack space A consists of all possible attack vectors, while K consists of known vulnerabilities. The mapping function $R : K \rightarrow A'$ translates these vulnerabilities into specific attack vectors, forming the reduced attack space A'.

- Reduction of Search Space: By definition, A' is constructed solely from the known vulnerabilities K, which implies A' is a subset of A. Thus, by construction, $|A'| \leq |A|$.
- Inside K-informed Space (A'): By focusing on K, attackers leverage historical data and known vulnerabilities, which are inherently more likely to be exploitable due to their established nature. Hence, the "density" of viable attack vectors within A' is higher, making the search more efficient and likely to yield fruitful results. Outside K-informed Space: While vulnerabilities can indeed exist outside of K, the absence of prior knowledge or evidence suggesting their exploitability means that the search within $A \setminus A'$ (the portion of A not included in A') is more akin to searching in the dark. The probability of discovering a new vulnerability in this area is lower, given the lack of targeted direction, making this effort less efficient and more time consuming.
- Probabilistic Advantage: The choice to operate within A' is not just about reducing the search space but also about maximizing the probability of finding exploitable vulnerabilities. The knowledge-driven approach inherently concentrates efforts where success is more likely, thus optimizing the search process.

□

Therefore, knowledge reuse effectively reduces the attack search space to a more manageable and potentially more fruitful subset A', where the efficiency of identifying new vulnerabilities is enhanced due to the higher density of known vulnerabilities. This strategic narrowing not only makes the attacker's job quicker but also more efficient, underlining the theorem's premise.

To further emphasize the effects of knowledge reuse on the attack search space, let us consider an illustrative example.

Example 7 (Impacts of Knowledge Reuse). Imagine an attacker has previously exploited 5 different systems. From these exploits, they have learned about 5 vulnerabilities, each corresponding to a specific attack vector. When faced with a new system, instead of considering every possible method of attack (the full search space A), they can narrow their focus to just these 5 known vulnerabilities.

So, even if the total number of potential attack vectors in the full search space A is, say, 1000, by relying on their previous knowledge, the attacker reduces their search space to just 5. This is a drastic reduction and showcases the efficiency gained through knowledge reuse.

6 Two Security Metrics

Now that we have a formal way of discussing knowledge and knowledge reuse in the context of computer security, we can now define new metrics in terms of knowledge and attacker search spaces.

Here we define two metrics, KOSM and DESM.

6.1 Knowledge Obfuscation Security Metric (KOSM)

The Knowledge Obfuscation Security Metric (KOSM) is designed to assess how well a system impedes attackers from effectively reapplying previously acquired knowledge. KOSM quantifies the effectiveness of a system's ability to obscure or distort previously acquired knowledge by potential attackers. By making previously gained knowledge unreliable or irrelevant, knowledge obfuscation reduces the value of attackers reusing known strategies or insights. A higher KOSM indicates that the system is better equipped at

rendering previously acquired attacker knowledge obsolete, thus increasing the cost and effort required for an attack.

Knowledge obfuscation can take many forms:

- Data Distortion: The defender intentionally modifies data to render them meaningless or misleading to an attacker who does not know how they have been transformed. Encryption is a kind of data distortion, as are transformations of data tables where columns and rows are renamed and/or rearranged.
- System Behavior Variation: The defender changes the behavior of a system over time or per interaction to ensure that attackers cannot reliably predict how the system will react. By adding unpredictability to the system's operations, attackers cannot effectively leverage their previous experiences or insights. Such variations could be arbitrary, say if the system randomizes how it responds to invalid input. Alternately, it could be adaptive, where the system explicitly responds differently when given abnormal (but technically legitimate) inputs.

Implicit in knowledge obfuscation is the understanding that defenders cannot really make their systems behave too differently over time if they wish their systems to function properly; however, defenders can periodically change how their systems behave so as to limit the utility of knowledge attackers have gained through reconnaissance, research, or attacks on other similar systems.

With this conceptual backdrop in place, we can now define KOSM precisely.

Definition 5 (**KOSM**). *Let us consider:*

- *K: The set of knowledge an attacker initially possesses about the system.*
- *K′: The perceived knowledge by the attacker after obfuscation strategies are applied.*
- *O: The obfuscation function that transforms K into K'.*

For the sake of quantification, let us associate a value with knowledge:

- *Value $V(K)$: The utility or effectiveness of the initial knowledge set K in terms of aiding an attack.*
- *Value $V(K')$: The utility or effectiveness of the obfuscated knowledge set K'.*

The effectiveness of KOSM can be represented by the reduction in knowledge utility:

$$\Delta V = V(K) - V(K') \tag{5}$$

Where:

- *A higher ΔV indicates a more effective knowledge obfuscation, implying that the system's security is enhanced by the obfuscation strategies.*
- *A ΔV close to zero would imply that the obfuscation strategies are not significantly impacting the utility of the attacker's knowledge, suggesting a need for stronger obfuscation measures.*

The KOSM metric is then defined as:

$$KOSM = \frac{\Delta V}{V(K)} \tag{6}$$

Where:

- *KOSM = 1 implies maximum obfuscation, rendering the attacker's prior knowledge completely useless.*
- *KOSM = 0 indicates no obfuscation, meaning the attacker's knowledge remains fully effective.*

To see the utility of KOSM, consider the following theorem.

Theorem 8. *For a system with applied KOSM, the expected number of successful attacks decreases as KOSM increases.*

Proof. Let us denote:

- $E_{initial}$: Expected number of successful attacks based on initial knowledge, K.
- $E_{obfuscated}$: Expected number of successful attacks based on obfuscated knowledge, K'.

From our earlier definition of KOSM:
$KOSM = \frac{\Delta V}{V(K)}$
Where:
$\Delta V = V(K) - V(K')$
Given that V(K) represents the utility or effectiveness of the knowledge set K for an attack, a higher KOSM value implies a greater reduction in knowledge utility.

Thus, as KOSM increases, the difference between $E_{initial}$ and $E_{obfuscated}$ becomes more pronounced:
$E_{initial} - E_{obfuscated} \propto KOSM$
Hence, $E_{obfuscated} < E_{initial}$ as KOSM increases, proving the theorem.

□

Note that what we refer to here as knowledge obfuscation overlaps with the concept of security through obscurity. Publicly available knowledge about a system's defenses may empower attackers to refine their strategies. However, by obfuscating key elements, defenders can mitigate the reuse of known vulnerabilities [23]. While security through obscurity has been criticized as a weak defense, it remains foundational in certain contexts, particularly in systems utilizing AI [1]. While security through obscurity refers to a specific state of a system (that some knowledge is obscure), knowledge obfuscation refers to the ability of attackers to make knowledge "obscure."

In summary, the KOSM metric quantifies the security of a system by measuring the reduction in the utility of attacker knowledge due to obfuscation strategies. A system with a high KOSM value effectively neutralizes the advantage an attacker would have from their prior knowledge without the defender having any specific knowledge of the exploits the attacker may be targeting.

6.2 Defense Evolution Security Metric (DESM)

The Dynamic Defense Evolution Security Metric (DESM) quantifies a system's ability to dynamically adapt and evolve its defenses in response to potential threats. A higher DESM value indicates the system's proficiency in altering its defenses to stay ahead of attackers. This continuous evolution ensures that the system's defenses

remain effective against the ever-evolving landscape of attacks. Primary facets of DESM include:

- Real-time Adaptability: The system's ability to adjust its defenses instantly based on detected threats or vulnerabilities. For example, upon detecting an abnormal surge in traffic, a system dynamically adjusts its firewall rules or deploys additional resources to mitigate a potential DDoS attack.
- Predictive Evolution: The system's capacity to anticipate potential future threats and adjust its defenses accordingly. For instance, leveraging AI and machine learning to analyze historical data and predict future attack patterns, then proactively adjusting defenses before these attacks materialize.

To further elucidate the concept of DESM, let us delve into its formal definition.

Definition 6 (**DESM**). *Let us consider:*

- *A: The initial set of attack vectors an attacker might employ.*
- *A': The set of attack vectors after the system has evolved its defenses.*
- *D: The dynamic defense function that transforms A into A'.*

For quantification, let us associate a success rate with attack vectors:

- *Success Rate $S(A)$: The probability of a successful attack using vectors in A.*
- *Success Rate $S(A')$: The probability of a successful attack using vectors in A' after defensive evolution.*

The effectiveness of DESM is represented by the reduction in attack success rate:

$$\Delta S = S(A) - S(A') \tag{7}$$

Where:

- *A higher ΔS indicates a more effective defense evolution, implying enhanced system security.*
- *A ΔS close to zero suggests that the dynamic defenses have not significantly impacted the effectiveness of the potential attack vectors.*

The DESM metric is then defined as:

$$DESM = \frac{\Delta S}{S(A)} \tag{8}$$

Where:

- *DESM = 1 implies that the defenses have evolved to render all prior attack vectors completely ineffective.*
- *DESM = 0 indicates no change in the effectiveness of the attack vectors, suggesting no significant defense evolution.*

The following theorem relates DESM to attacker successes.

Theorem 9. *For a system with applied DESM, the time to a successful attack increases as DESM increases.*

Following our theorem's proposition, we now present a proof, elucidating the conditions under which one strategy outshines the other.

Proof. Let us denote:

- $T_{initial}$: Time taken for a successful attack based on initial attack vectors, A.
- $T_{evolved}$: Time taken for a successful attack based on evolved attack vectors, A'.

From our earlier definition of DESM:
$DESM = \frac{\Delta S}{S(A)}$
Where:
$\Delta S = S(A) - S(A')$
Given that S(A) represents the success rate of an attack using vectors in A, a higher DESM value implies a greater reduction in attack success rate.

Now, if the success rate of an attack decreases, it implies that an attacker would need to invest more time to achieve a successful breach.

Thus, as DESM increases, the difference between $T_{evolved}$ and $T_{initial}$ becomes more pronounced:
$T_{evolved} - T_{initial} \propto DESM$
Hence, $T_{evolved} > T_{initial}$ as DESM increases, proving the theorem.

□

To summarize, KOSM attempts to capture the power of obfuscation as a strategy for minimizing the value of an attacker's gathered intelligence, while DESM models the potential benefits of defenders continuously evolving their defenses.

7 KOSM & DESM in Practice

On their own, KOSM and DESM cannot be used to measure the absolute or relative security of real-world systems because they are defined in very abstract terms. The utility of these metrics, however, come in how they allow us to conceptualize the larger relationships between attack strategies, defensive strategies, and security metrics.

To better understand these metrics and their implications, we now discuss potential scenarios and relate them to KOSM and DESM, respectively.

7.1 Applying KOSM

Our Knowledge Obfuscation Security Metric, KOSM, measures the ability of defenders to reduce the value of knowledge attackers have acquired by transforming their systems in some way, ideally through automated means. Intuitively KOSM would seem to directly measure the entropy introduced by defenses such as address space randomization [21] and other moving target defenses [6]. However, that is not quite what KOSM captures.

Consider the case of address space layout randomization, or ASLR. When it was was first introduced, a system with ASLR would stop virtually all attempted exploits of buffer overflow vulnerabilities because processes on protected systems would not have predictable memory layouts. However, once ASLR became widespread, attackers developed techniques such as heap spraying [17] that enabled code injection attacks even when memory layout was not consistent. These methods are significantly more complex than classic buffer overflow attacks and they don't work in all circumstances; however, when they do, a buffer overflow vulnerability

can be just as devastating as they were before ASLR defenses were introduced.

Thus, when ASLR was introduced, it had a KOSM value of close to one for buffer overflow-type vulnerabilities, because the obfuscation (ASLR) made knowledge of a class of vulnerabilities essentially useless as they couldn't be exploited. However, as attackers developed ways to exploit buffer overflows even in the presence of ASLR, the KOSM value of ASLR has decreased significantly, because now knowledge of a buffer flow retains value even in the presence of ASLR. In this context KOSM for ASLR is not zero, but it is no longer close to one.

We expect similar KOSM trajectories for most defenses: when first introduced, they invalidate entire categories of attacker knowledge through relatively constrained system-level changes. As attackers develop countermeasures, however, KOSM will decrease. For most defenses KOSM will not ever go to zero, as the defense will still serve to mitigate the exploitation of some vulnerabilities; without further defender innovation, however, the value of any static defense will tend to decrease to the point that it provides minimal protection against new attacks, no matter how much benefit it provided when it was first introduced. KOSM thus captures how the advantage provided by obfuscation-style defenses (those that mitigate vulnerabilities without removing them) tends to go down as attackers develop countermeasures. This attacker evolution necessitates ongoing defender evolution, something that we model in part with DESM.

7.2 Applying DESM

While KOSM captures the relative advantage an obfuscation defense gives in mitigating vulnerabilities, Dynamic Defense Evolution Security Metric (DESM) captures to what degree a defense can protect against a set of attacks. Note that a given vulnerability can be exploited by many different attack vectors: a single buffer overflow vulnerability could be exploited by 100 different pieces of malware. Thus, attacker knowledge of vulnerabilities is not in the same conceptual space as defender knowledge about specific attack vectors. To better understand DESM, here we examine DESM in the context of some standard dynamic defenses.

First, consider signature-based anti-malware defenses. We can think of A as representing the set of malware that can successfully compromise a system, and A' as the set of malware that can compromise a system after a signature update. If no signatures were added, then A and A' will be equal and DESM will be zero, meaning that no dynamic defense evolution has taken place. Similarly, if the defenders add a signature that blocks one piece of malware and the set of malware targeting a system is relatively large, then DESM will be very close to zero even after a signature has been added—the system's defenses have adapted but not by very much.

Next, consider automated software updates. An update's impact on DESM will vary widely. At one end, some updates will have a DESM of zero simply because the updates add or change functionality. Updates that patch specific security vulnerabilities will generally have a non-zero DESM, but the degree of change is proportional to the number of impacted attack vectors, i.e., how much malware has been thwarted by patching a specific vulnerability. Some updates patch vulnerabilities that have not been targeted while others patch vulnerabilities that have been widely used. A patch strategy intended to maximize DESM would thus focus on patching vulnerabilities that are more likely to be targeted by attackers.

DESM can also be applied to defense strategies that change the behavior of people rather than computer systems. For example, by quantifying employees' willingness to click on email links and disclose confidential information through insecure channels, organizations can quantify their susceptibility to phishing attacks. They can then design training programs and evaluate their effectiveness by how they impact employee susceptibility metrics [9]. Organizations that use such training metrics and who continuously update their curriculum in response to new threats can be said to have a high DESM for phishing, while organizations that do not monitor employee performance or who do not update their training materials would have a lower DESM for phishing.

Last, consider anomaly-based intrusion detection systems that detect attacks by learning models of normal system behavior and responding to deviations from those models. Because such systems can be bypassed by attackers mimicking normal system behavior and such normal behavior can vary widely between systems, anomaly-based intrusion detection systems cannot provide guaranteed protection against any malware or exploits of any vulnerability. However, because they can detect with some probability a wide variety of attacks, adding them as a defense layer increases the DESM of a system, potentially significantly. Further, this DESM level can be maintained to a degree even as attackers change their attack vectors, even if the defenders make no further changes.

Anomaly detection systems are general defenses that provide some protection in a wide variety of circumstances while signature based anti-malware defenses and automated updates provide a great deal of protection in very specific cases—and almost no protection otherwise. By capturing the ability of defenders to respond to changing attack vectors, DESM gives us a way of distinguishing between these defensive strategies.

8 Discussion

Our key contributions here are 1) the insight that knowledge reuse can be used as a foundation for security metrics, 2) search spaces, as used in AI, can be used to formalize knowledge reuse in security, 3) a formal model of knowledge reuse in security, and 4) two metrics, KOSM and DESM, that capture different aspects of knowledge reuse dynamics between attackers and defenders.

The key advantage of our formal yet abstract approach to the problem of security metrics is that it allows us to think about the relationship between attackers and defenders without reference to specific attacks or technologies, allowing us to instead focus on the dynamics of this interaction. It is well known that despite ongoing investments in computer security, attackers retain an ongoing advantage in that systems large and small are compromised on a regular basis. Our new metrics can give us some insight as to why.

KOSM can help us think about how the advantages of specific static defenses decrease over time, while DESM can help capture a defender's ability to respond to new threats. To be sure, applying either metric to any specific situation can be difficult, as it requires conceptualizing attacker and defender moves in a dynamic, abstract

search space of variable size. We hypothesize, though, that the very act of thinking in this way can help us better understand defenses from a strategic perspective, helping security researchers and practitioners to better understand the dynamics of attacker knowledge reuse. Under what circumstances does randomization improve security? What is the value of defenses that provide only partial protection but can do so against a wide variety of threats? When are evasion and mimicry attacks worth developing countermeasures against? These strategic questions are difficult to conceptualize in most security models yet are relatively easy to model using our attack difficulty formalism.

The mantra "no security through obscurity" has often been used to minimize the value of various defense strategies. Perhaps the key insight of this work is that obscurity is actually a foundational security strategy, one that we must better understand how and when to employ if we wish to materially change the balance of power between attackers and defenders. We hope this work will help motivate others to work more on these foundational problems.

References

[1] Ross J Anderson. *Security engineering: a guide to building dependable distributed systems*. John Wiley & Sons, 2010.

[2] Max Boisot and Agustí Canals. Data, information and knowledge: have we got it right? *Journal of Evolutionary Economics*, 14:43–67, 2004.

[3] Cormac Herley and Paul C Van Oorschot. SoK: Science, security and the elusive goal of security as a scientific pursuit. In *2017 IEEE Symposium on Security and Privacy*, pages 99–120. IEEE, 2017.

[4] Cormac Herley and Paul C Van Oorschot. Science of security: Combining theory and measurement to reflect the observable. *IEEE Security & Privacy*, 16(1):12–22, 2018.

[5] Debra S Herrmann. *Complete Guide to Security and Privacy Metrics: Measuring Regulatory Compliance, Operational Resilience, and ROI*. Auerbach Publications, New York, 2007.

[6] Sushil Jajodia, Anup K Ghosh, Vipin Swarup, Cliff Wang, and X Sean Wang, editors. *Moving Target Defense: Creating Asymmetric Uncertainty for Cyber Threats*, volume 54 of *Advances in Information Security*. Springer, New York, NY, 2011.

[7] Sushil Jajodia, Steven Noel, and Brian O'Berry. Topological analysis of network attack vulnerability. *Managing Cyber Threats: Issues, Approaches, and Challenges*, pages 247–266, 2005.

[8] Andrew Jaquith. *Security Metrics: Replacing Fear, Uncertainty, and Doubt*. Pearson Education, 2007.

[9] Matthew L Jensen, Michael Dinger, Ryan T Wright, and Jason Bennett Thatcher. Training to mitigate phishing attacks using mindfulness techniques. *Journal of Management Information Systems*, 34(2):597–626, 2017.

[10] Susan Landau. Standing the test of time: The data encryption standard. *Notices of the AMS*, 47(3):341–349, March 2000.

[11] Saran Neti, Anil Somayaji, and Michael E Locasto. Software diversity: Security, entropy and game theory. In *7th USENIX Workshop on Hot Topics in Security (HotSec 12)*, Bellevue, WA, August 2012. USENIX Association.

[12] Oluwatosin Ogundare. How do you know what you know: Epistemology in software engineering. *Journal of Software Engineering and Applications*, 10(2):168–173, February 2017.

[13] Shirley C Payne. A guide to security metrics. *SANS Institute Information Security Reading Room*, 2006.

[14] Marcus Pendleton, Richard Garcia-Lebron, Jin-Hee Cho, and Shouhuai Xu. A survey on systems security metrics. *ACM Computing Surveys (CSUR)*, 49(4):1–35, December 2016.

[15] Wolter Pieters, Dina Hadziosmanovic, Aleksandr Lenin, Lorena Montoya, and Jan Willemson. TREsPASS: Plug-and-play attacker profiles for security risk analysis. *35th IEEE Symposium on Security & Privacy, Poster Abstracts*, 2014.

[16] David L Poole and Alan K Mackworth. *Artificial Intelligence: Foundations of Computational Agents*. Cambridge University Press, New York, NY, 2010.

[17] Paruj Ratanaworabhan, V Benjamin Livshits, and Benjamin G Zorn. NOZZLE: A defense against heap-spraying code injection attacks. In *18th USENIX Security Symposium*, pages 169–186, 2009.

[18] Dirk Rijmenants. One-time pad, 2020. Last retrieved November 1, 2024. Original: http://users.telenet.be/d.rijmenants/en/onetimepad.htm. Current: http://hawkgirl.net/documents/communication/One-time-pad.pdf.

[19] Sankardas Roy, Charles Ellis, Sajjan Shiva, Dipankar Dasgupta, Vivek Shandilya, and Qishi Wu. A survey of game theory as applied to network security. In *2010 43rd Hawaii International Conference on System Sciences (HICSS '10)*, pages 1–10. IEEE, 2010.

[20] S.J. Russell, P. Norvig, and E. Davis. *Artificial Intelligence: A Modern Approach*. Prentice Hall series in artificial intelligence. Prentice Hall, 2010.

[21] Hovav Shacham, Matthew Page, Ben Pfaff, Eu-Jin Goh, Nagendra Modadugu, and Dan Boneh. On the effectiveness of address-space randomization. In *Proceedings of the 11th ACM Conference on Computer and Communications Security*, pages 298–307, 2004.

[22] Oleg Sheyner, Joshua Haines, Somesh Jha, Richard Lippmann, and Jeannette M Wing. Automated generation and analysis of attack graphs. In *2002 IEEE Symposium on Security and Privacy*, pages 273–284. IEEE, 2002.

[23] Adam Shostack. *Threat Modeling: Designing for Security*. John Wiley & Sons, 2014.

[24] Jonathan M Spring, Tyler Moore, and David Pym. Practicing a science of security: A philosophy of science perspective. In *Proceedings of the 2017 New Security Paradigms Workshop*, pages 1–18, 2017.

[25] Richard S Sutton and Andrew G Barto. *Reinforcement Learning: An Introduction*. MIT press, 2018.

[26] Haridimos Tsoukas. *Complex Knowledge: Studies in Organizational Epistemology*. Oxford University Press, 2004.

[27] Vilhelm Verendel. Quantified security is a weak hypothesis: A critical survey of results and assumptions. In *Proceedings of the 2009 New Security Paradigms Workshop*, pages 37–50, 2009.

[28] David H Wolpert, William G Macready, et al. No free lunch theorems for search. Technical Report SFI-TR-95-02-010, Santa Fe Institute, February 1996.

[29] George O.M. Yee. Security metrics: An introduction and literature review. *Computer and Information Security Handbook*, 2013.

[30] Yin Zhang and Vern Paxson. Detecting stepping stones. In *USENIX Security Symposium*, Denver, Colorado, August 2000.