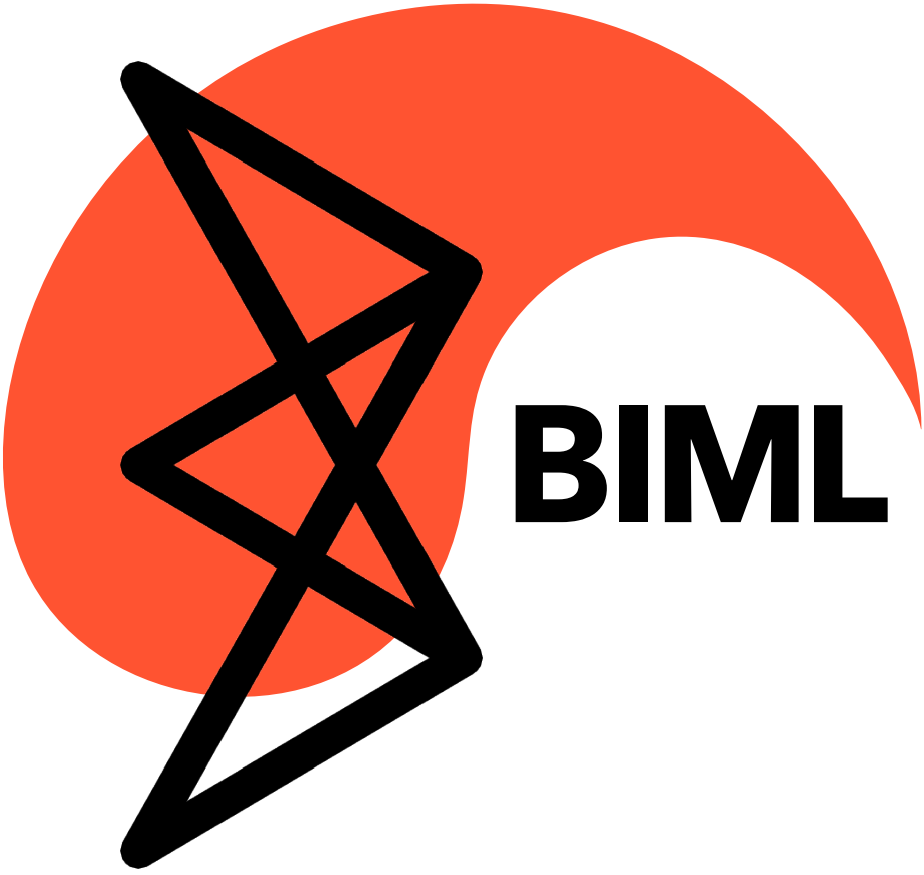
NO SECURITY METER FOR AI

Gary McGraw, Ph.D.
Harold Figueroa
Katie McMahon
Richie Bonett



Berryville Institute of Machine Learning (BIML)
https://berryvilleiml.com
Version 1.0 | May 13, 2026

Defining some Terms

Before we dive in, a little bit about nomenclature (which has always been a challenge, both in security engineering and in AI/ML).

When we say ***AI***, we really mean ***machine learning (ML)***. We think of ML systems as *what machines* that learn their behavior from immense WHAT piles (as opposed to HOW *machines* that are specified by a detailed program).

Agentic AI enhances an ML model with tools and sensors that connect it to the world in an action loop. (In some sense Agentic harnesses are like perception and action apparatus.)

We define ***security*** as the process of engineering systems to ensure they continue to function correctly under malicious attacks. This involves building security into a system during its development, rather than just protecting it after it has been created.[4]

A ***risk*** is the probability that an asset will suffer an event of a given negative impact, determined by such factors as the ease of executing an attack, the motivation of an attacker, and the existence of vulnerabilities in a system.

By contrast, a ***threat*** is an actor or agent exploiting a risk.

Finally, we call the nascent field of AI security engineering ***Machine Learning Security (MLsec)***.

Executive Summary

Let's say you wanted to make sure that your AI is secure. Can you just maximize the security and privacy benchmark and call it a day? Nope, because benchmarks don't actually work for measuring AI capabilities (even when they are NOT emergent systemic properties like security). So let's take a step back: how do you measure security in the first place? Good question. Over the last 30 years, security engineering for software evolved from black box penetration testing, through whitebox code analysis and architectural risk analysis to de facto process-driven standards like the Building Security In Maturity Model (BSIMM). Software had a very deep impact on business operations, and it appears that AI is going to have an even deeper impact. Will a software security-like measurement move work for AI? Probably. In the meantime we can make real progress in AI security by cleaning up our WHAT piles and managing risk by identifying and applying good assurance processes. (Spoiler alert: no matter what we do, we still don't get a security meter for AI, so we need to be extra vigilant about security.)

BIML

Introduction

One of the major challenges to the emergence of machine learning security as a discipline is the disconnect between AI/ML people, who tend to know very little about security engineering (and who may care even less) and system-level security engineers who are not usually familiar with the off-nominal failure conditions of ML/AI as a system component. This disconnect leaves lots of room for security failure to rear its ugly head, often in novel and amusing ways.

Engineering very often turns to measurement as a useful technique in this kind of possible-failure situation.[1] Want to predict failure in cargo shipping? Better figure out how to measure the tensile strength of various metals, how to measure ship hull architecture, and how to measure and predict weather patterns, etc. Given enough time and data, scientists can determine measurements that fit the bill, leaving a clear path for armies of actuaries to build a world model fit for insurance-based risk transfer. *Et voila*—marine insurance.

When it comes to machine learning security, then, one of the first orders of business is to figure out a set of measurement schemes to use; something that seems easy enough on the face of it.

That's where the disconnect between ML people and security engineers is causing some real trouble. We aim to bridge that disconnect in this article, describing first how AI benchmarks work (or, don't work as the case may be), and second, the history of measurement in software security (itself ending on a rather unsatisfying note). Our goal is to lay bare a set of ideas that both sides can understand, so that together we might make some forward progress towards security measurement for AI.

If you find yourself approaching this article strictly in one camp or the other, please bear with us as we talk about your camp. The connection is where the white space is. If, on the other hand, you don't think security concerns will have much impact on the rapid adoption of AI technology, maybe this article will change your mind.

Preamble

The world loves a good security meter: black box testing by "reformed" hackers, white-hat penetration testing results, "military grade cryptography" letter grades, and even AI performance over benchmarks called SECURE[2] and PRIV-IQ[3]. The problem is that none of these things work as security meters.

Measuring security is non-trivial, because security is a system-wide emergent property. As a result, when it comes to measuring AI security we have two obstacles to overcome—measurement in AI and measurement in security engineering.

Measuring AI: Why benchmarks are broken

AI has a measurement problem.

When large datasets were finally gathered and cataloged (or tagged and bagged, as the case may be) by pioneers in the field like Fei-Fei Li who assembled ImageNet[5] in 2006, the most obvious measurement approach was to use the dataset itself as a natural born benchmark. (ImageNet was a large-scale visual database with over 14 million labeled images across thousands of categories.) The Convolutional Neural Network, made famous by Yann LeCun was a huge AI breakthrough when it achieved a top-5 error rate of 15.3% over ImageNet in 2012.[6] The basic approach to measurement spearheaded by ImageNet worked for a while when ML progress was incremental and

task scope was clear, network size was smallish, and there was a clear way to measure system performance over a properly delineated dataset (often a held-back evaluation set not used directly in training).

Things in AI measurement changed for the worse with the post 2018 advent of LLMs based on Vaswani's transformer architecture.[7] Models grew, datasets grew, and scaling "laws" suggested that models generalize well given enough data.[8,9] Datasets got so big that tagging and bagging everything became (and remains) impossible. Instead of supervised learning, auto-association in a prediction task was used for training and measurement. Suddenly it made much less sense to use a held-back evaluation set as a benchmark.

Instead, the nature of benchmarks completely evolved, from held-back evaluation to quizzes, things that looked much more like "tests humans have to take" (including the LSAT), and scenario tests. This progression, which Melanie Mitchell describes as moving from "measuring perception to measuring cognition," happened in three general waves as task scope expanded and became unbounded. GLUE (2018) introduced the idea of basic "understanding" around whether one sentence entails another. Next came the "human test" wave best characterized by the MMLU (Massive Multitask Language Understanding)[10] benchmark in 2020. The third wave, starting in 2023, is still underway and involves observing Agentic AI in various scenarios.

New benchmarks continue to proliferate like mushrooms after a spring rain. They have become unsettling and uncanny—Halloween costumes in a security theater. And just to confuse things further, these benchmarks are named things like HumanEval (for coding tasks)[11], ARC (AI2 Reasoning Challenge)[12], GSM8K (Grade School Math)[13], and HellaSwag (for commonsense reasoning)[14], leading non-scientists to believe that maximizing performance on a given benchmark, say the SECURE benchmark suite we mentioned above, is tantamount to being secure.

Note that this isn't just a problem for security and privacy in AI, it applies to the way we measure everything in AI today. There is a serious and obvious misalignment in the assumptions benchmarks are built and used around.

Nonetheless, benchmarks proliferated and leaderboards like Open LLM Leaderboard[15], Scale SEAL Leaderboard[16], Vellum.ai Leaderboard[17], and Chatbot Arena[18] took hold. These days, it is all too common for a research "paper" posted on *arXiv* to introduce a new benchmark specially designed to make the research look great. Naming the novel benchmark something cool is *de rigeur.* (Science papers used to be peer reviewed and in some sense stress tested. That was the science part. Today, *arXiv* has exploded in volume with no approval process other than being picked up and read by interested bystanders.)

One of the big philosophical problems with AI benchmarks harkens way back to the ELIZA effect—that is anthropomorphism. Simply put, today's AI models based on LLMs do not think like people do and we should try to avoid talking about them as if they do. They don't have "minds" like we do. They are, instead, "predictive interpolation engines" without a stable internal world model.[19] So the question is, how to measure these things effectively.

Mitchell directly targets this problem in her NeurIPS keynote (delivered December 4, 2025 in Vancouver) where she introduces six principles to guide cognitive evaluation of AI models (treating them as "alien intelligences").[20] As Mitchell describes, beating a human on an arbitrary human benchmark (like, say, the LSAT) is not very meaningful. Here are the six principles as presented in her work on cognitive evaluation:

> *Principle 1: Be aware of your own anthropomorphic cognitive biases.*
>
> *Principle 2: Be skeptical of others' (and your own) hypotheses. Design control experiments for possible alternate strategies that could produce the observed behavior.*
>
> *Principle 3: Design novel variations of stimuli or benchmark items to test robustness and generalization.*
>
> *Principle 4: Be curious about mechanisms underlying performance.*
>
> *Principle 5: Consider performance versus competence.*
>
> *Principle 6: Analyze failure types, and embrace "negative" results.*

Mitchell's principles cut directly against the way benchmarks (including security benchmarks) are currently used in AI. Principle 1 warns against the anthropomorphic assumption that a model scoring well on a human-readable security scenario is actually, itself, "doing security." Principle 4 warns us precisely where external benchmarks fall down: they tell us what a model does, not why or how. Principle 5, performance versus competence, maps cleanly onto our strange loop problem: a model may perform well on a security task without possessing anything like genuine security competence (and an attacker may well find the difference). Taken together, Mitchell's framework suggests that the alien intelligence view of AI measurement is not just a philosophical nicety—it has direct practical consequences for anyone tempted to use a benchmark score as a security rating.

Attempts at Security Benchmarking with AI

Given the alien intelligence view of measurement most notably represented by Mitchell, let's take a deeper look at SECURE and PRIV-IQ and determine what they might actually measure. (They do, we will admit, have very compelling names.). We'll focus on these new Agentic AI measurement schemes in order to attempt to be up to date.

The SECURE (Security Extraction, Understanding & Reasoning Evaluation) benchmark introduced by RIT in 2024 is designed to assess ML performance "in realistic cybersecurity scenarios" around SOCs, CERTs, and threat-intel teams. We find these scenarios act as little stand alone security puzzles that an AI is supposed to solve, with not much to say about the security capabilities of the models. SECURE and closely-related benchmarks including the Cybersecurity AI Benchmark (CAIBench)[21] and Microsoft's ExCyTIn (Executive Cyber Threat Investigation)[22] are all static fixed tests. That means when an ML model runs against the test, it is being probed in a specific, simulated scenarios. And it also means that the tests, once published, directly pollute results (because they are hoovered up into training sets).

The contamination problem is compounded by a more alarming one. Recent work from UC Berkeley demonstrates that eight of the most prominent AI agent benchmarks (including SWE-bench and WebArena) can be exploited to achieve near-perfect scores without solving a single task.[23] No reasoning, no capability—just exploitation of how the score is computed. If general-purpose agent benchmarks are this fragile, security benchmarks, which face the same structural problems, are unlikely to fare better.

Comparing different models with these benchmarks is possible. Here is one such table generated by Gemini under our direction in March 2026. This is what we mean when we reference a leaderboard approach. DISCLAIMER: the leaderboard shown below was created by an AI model for demonstration purposes only and is wrong in many ways.

Model	RIT SECURE (ICS/Critical Infrastructure)	CAIBench (Offensive/Defensive)	ExCyTIn (SOC Investigation Reward out of 1)
GPT-5 (High Reasoning)	92.4%	44.2%	0.368
Claude 4.5 Sonnet	89.1%	41.5%	0.334
Gemini 2.5 Pro	88.5%	38.8%	0.315
Llama 4 (Scout)	84.6%	35.2%	0.290
o3-mini (Reasoning)	81.2%	39.7%	0.296
Kimi K2.5 (Thinker)	83.3%	32.1%	0.255
WARNING: *this example leaderboard was generated by AI and is wrong for any number of reasons.*			

A Software Security Side Bar

Standard security benchmarks—including the Juliet Test Suite [1], the OWASP Benchmark [2], and curated vulnerability datasets—have served as the primary evaluation methodology for static analysis tools for more than a decade. However, for LLM-based vulnerability detection systems, these benchmarks introduce significant methodological concerns.

The training corpora of large language models contain vast quantities of publicly available source code, technical documentation, and security research. Benchmarks such as Juliet, OWASP Benchmark, and many publicly reproduced CVE examples are openly available and therefore likely to appear in the training data of modern code-oriented LLMs. Under these conditions, high benchmark accuracy may reflect memorization rather than genuine vulnerability reasoning. This concern has been demonstrated empirically: studies of LLM benchmark leakage show that models can achieve dramatically higher performance on contaminated benchmark samples compared to unseen data. For example, recent analysis across software engineering benchmarks reports pass rates up to 4.9× higher on leaked samples compared to non-leaked ones [3]. Broader surveys of contamination in LLM evaluation emphasize that benchmark validity requires explicit contamination analysis when models may have been exposed to the benchmark during training [4,5].

The reliability of existing vulnerability datasets is further complicated by data quality issues. An empirical study of commonly used vulnerability datasets found that 20–71% of vulnerability labels were inaccurate and 17–99% of samples were duplicated, raising concerns about their suitability for evaluating machine learning models [6]. Subsequent work introducing the PrimeVul dataset, which focuses on realistic vulnerabilities extracted from real repositories, reports that LLM-based detectors perform substantially worse on realistic vulnerability detection tasks than on synthetic benchmarks. In particular, detection accuracy on synthetic datasets such as Juliet was observed to be 10.5% higher on average than on real-world datasets [7,8].

Additional characteristics of synthetic benchmarks further limit their suitability for evaluating LLM-based systems. For example, the Juliet Test Suite contains over 80,000 test cases with systematic naming conventions (e.g., CWE89_SQL_Injection__*) that reveal the vulnerability type directly in the filename. Similar patterns exist in OWASP Benchmark test cases. These artifacts enable models to infer the vulnerability category without analyzing the code itself, making the benchmarks vulnerable to shortcut learning.

Because benchmark datasets may both appear in model training data and contain structural artifacts, evaluation based solely on benchmark performance cannot reliably measure a model's vulnerability reasoning capability.

[1] National Institute of Standards and Technology, "Software Assurance Reference Dataset (SARD): Juliet Test Suite." [Online]. Available: https://samate.nist.gov/SARD/.

[2] OWASP Foundation, "OWASP Benchmark Project." [Online]. Available: https://owasp.org/www-project-benchmark/.

[3] D. Guo et al., "LessLeak-Bench: A First Investigation of Data Leakage in LLMs Across 83 Software Engineering Benchmarks," arXiv:2502.06215, 2025.

[4] O. Sainz, J. A. Campos, I. Garcia-Ferrero et al., "NLP Evaluation in Trouble: On the Need to Measure LLM Data Contamination for Each Benchmark," in *Findings of the Association for Computational Linguistics: EMNLP 2023*, 2023.

[5] Y. Li et al., "Unveiling the Spectrum of Data Contamination in Language Models: A Survey from Detection to Remediation," in *Findings of the Association for Computational Linguistics (ACL 2024)*, 2024.

[6] R. Croft, M. A. Babar, and M. M. Kholoosi, "Data Quality for Software Vulnerability Datasets," in *Proc. 45th Int. Conf. Software Engineering (ICSE)*, 2023.

[7] Y. Ding, Y. Fu, O. Ibrahim et al., "Vulnerability Detection with Code Language Models: How Far Are We?" in *Proc. 47th Int. Conf. Software Engineering (ICSE)*, 2025.

[8] X. Yang et al., "Understanding the Effectiveness of Large Language Models in Detecting Security Vulnerabilities," arXiv:2311.16169, 2023.

Just to be super clear here, SECURE is focused on an abstract recognition task from security operations—diagnosing security problems in industrial control systems—and attempts to probe how well known vulnerabilities can be recognized and extracted from MITRE's *Attack* and *Common Weakness* datasets to solve problems posed by "real-world" scenarios. CAIBench also focuses on security operations, using Docker containers to drop AI agents into attack and defense scenarios—a move that purports to measure multi- step adversarial reasoning. (Note that this hypothetically requires a mental model of an evolving dynamic environment, something that may interest the AI researchers reading this.) Finally, ExCyTIn immerses agents in a simulated Azure Sentinel SOC environment (obviously rooted in security operations again), with 44 days of logs: meant to probe investigative reasoning on a threat hunt.

To belabor our point for clarity purposes, each of these benchmarks rests on key defining assumptions: for SECURE, the weaknesses database the AI is working from is assumed to be complete; the CAIBench domain is a toy domain, like a game; and, ExCyTIn assumes that the logs are sufficient for a threat hunt. These are all important limits which serve to close and formalize what otherwise might be thought of

as open systems. Further, these security benchmarks measure how well a tool does in a security situation, but don't say anything about the security of the tool itself.

Obviously, we are well past the days of directly probing a model's training data (through a held back set of probes looking for perceptual generalization)! We have indeed made the move towards cognition, trying to assess a model's behavior in a complex environment. But some problems remain for security. Most notably: 1) passing a specific toy scenario is no guarantee of general ability (especially if the scenario has been widely published), 2) new attacks and vulnerabilities are released every day, and 3) it's easy to misuse the benchmark score as a security rating.

It's this last one that we want to harp on. Using a benchmark score as a security rating (say, 92.4% secure) seems an awful lot like an annual pen test in a trench coat because *it deeply misconstrues what exactly is being tested here*. Performance on a particular test does not always correlate with resilience in the wild. More specifically to AI, models can perform well on various probes of security principles nested inside scenarios but still go wildly off the rails when confronted with much messier real world situations, especially since they are trained to complete everything as helpfully as possible by their very design. (Ironically, security is not often accused of being helpful or of actually helping to get things done.)

The yin/yang nature of security is a challenge as well. Identifying problems during operations of a system is one thing. But designing and building systems that don't suffer from these problems is something else entirely. This is a systemic problem in the field of security that's being directly reflected in (or rather, trained into) our ML systems.

Security is a Strange Loop

Finally, and most importantly of all when it comes to AI benchmarks: **an AI system that performs well on a security benchmark may itself not be secure.** That's the key difference between Machine Learning Security (building ML systems that ARE secure) and Using Machine Learning to DO security (ML systems used FOR security). This is all very confusing, and confusing for both ML practitioners who may believe security is one tangible self-referential field and security people who may believe that an ML system that can perform well on a battery of security tests is itself secure.

The best analogy here comes from the field of software security, where the main objective is building secure software. Using the processes and tools developed for software security, we can sometimes build software that is actually secure. Here comes the kicker: security software is software used FOR security (for example Norton anti-virus software). It is too often the case in the real world that security software suffers from very bad software security. In the best of all possible worlds, security software would be built following software security best practices and all this tricky word ordering would no longer matter.

Before moving on, it is worth noting that privacy suffers from exactly the same measurement problem—and we can dispatch it quickly. Just as we did for security, we can build a leaderboard for AI privacy, using a set of benchmarks. Once again, this leaderboard was generated by Gemini under our direction in March 2026. DISCLAIMER: the leaderboard shown below was created by an AI model for demonstration purposes only and is wrong in many ways.

Model	Overall Priv-IQ Score	Visual Privacy (PII in Images)	Legal Logic (Global Regulations)	Data Anonymization (Agentic Masking)
GPT-5 (High Reasoning)	88.4%	85.2%	94.1%	86.0%
Claude 4.5 Sonnet	86.2%	82.1%	91.5%	85.1%
Gemini 2.5 Pro	84.7%	89.4%	88.5%	76.2%
Llama 4 (Scout)	79.1%	72.4%	84.6%	80.3%
o3-mini (Reasoning)	77.8%	70.2%	81.2%	82.0%
Kimi K2.5 (Thinker)	74.5%	78.3%	72.1%	73.1%
WARNING: *this example leaderboard was generated by AI and is wrong for any number of reasons.*				

Let's dig into the four privacy benchmarks the same way we dug into the security benchmarks above. Priv-IQ treats privacy as a multimodal capability encompassing visual and text-based data, with a domain focus on legal information. The benchmark encompasses a number of privacy tasks. Visual Privacy is probed by a dataset and associated benchmark, HR-VISPR[24]—a visual language model benchmark focused on detection of biometrics, ID documents, and other PII. The Legal Logic benchmark, PrivaCI-Bench[25], probes how a model uses the idea of contextual integrity as a foil for GDPR and CCPA. Finally, a study of Data Anonymization[26] in the healthcare domain shows how data utility trades off against various anonymization methods.

Unsurprisingly, the privacy case lands us in exactly the same position. The benchmarks measure privacy task performance, not whether the system is itself privacy-preserving. Same loop, different label.

The bottom line is this: not only do we have a problem measuring security in AI (since security benchmarks are not meaningfully correlated with system properties), we have a huge problem with AI benchmarks of every sort and need to come up with a better way of approaching this.

Agentic AI might offer us a way out of this situation by describing systems that include ML in terms of intention. If we know what an Agentic system "intends" to do (as a system with emergent behavior, not just as the union of all agent capabilities), perhaps we can impose some more meaningful security control.

Part of the solution may be to "read the minds" of ML systems by getting inside them and seeing how they work. If we can identify certain behavior patterns and correlate them with observable internal states, maybe we can leverage those insights in controls. (This will in all likelihood include modeling of "self" in the world, and developing a theory of mind.)

A Retrospective: Measuring software security from black box to BSIMM

Now that we have covered some basic issues in AI benchmarking and their impact on security measurement in AI, we will turn our attention to the more general problem of measuring security. We will use software security measurement as an explanatory foil.

In the mid 1990s, some people used to think that talking about security vulnerabilities and exploits was too dangerous. They discouraged publishing details about attacks or collecting databases of vulnerabilities. This began to

change with the proliferation of Bugtraq (starting in 1993) and the publication of *Exploiting Software*[27] (in 2004). To this day, there is still debate in the security community over "responsible disclosure," but commercial vulnerability marketplaces like HackerOne's bug bounty program are in full swing.

The idea of testing a system by breaking into it (or exploiting it) has a long history. One of the first systems to automate this philosophy was Dan Farmer's 1995 SATAN program, released with a paper entitled, "Improving the security of your site by breaking into it."[28]

The idea behind SATAN is both simple and economical—use a canned set of tests to determine whether or not your system configuration is secure. This makes all kinds of sense from an economic perspective since once you have a can of tests, all you need to do is run it as a set of magic regression tests all over the place and see what happens.

The problem here is not whether or not this works (it does), but what it actually measures. SATAN measures insecurity (or badness). It does not measure security. To get to the heart of this issue philosophically, BIML's McGraw coined the term "badness-ometer" in 2006.[29]

Bottom line: penetration testing from the outside➔in is effective as a badness-ometer, especially when it is carried out by knowledgeable and crafty security engineers (aka hackers). But a badness-ometer simply measures a range from you're in "DEEP TROUBLE" (because you fell prey to our can of tests) at one end to "WHO KNOWS" (you passed all of the known tests that we put in that can) at the other (see figure above). As it turns out, penetration testing is not effective at all as a security meter.

One redeeming factor about penetration testing is that most of the time when we find an actual exploit we know how to fix it. This is easiest when the problem is configuration related. Using the wrong version of something? Update it. But it gets much murkier when we progress to black box probing of system internals. We know there is a problem in there somewhere, and we can leverage it from outside into a working exploit, but we are not sure exactly where it is in there, and thus have a hard time fixing it properly?!

Opening things up to whitebox probing helps with the observability problem. In the field of software security, for example, we even figured out how to look for bugs before code was compiled, using static analysis.

Step one in building a more effective security approach is to get inside the system while you are designing and implementing it. We can apply our notion of a badness-ometer at every level: outside➔in, inside itself looking at a particular implementation for bugs, and inside looking at design for flaws. Thing is, none of these testing-based approaches gets us past the badness-ometer philosophy. Badness-ometers are great during engineering, but they only allow us to find and fix problems.

Software security has addressed this issue in tech by identifying and measuring assurance activities that theoretically result in better, more secure software. (That's just a fancy way of saying process.) When we say we want to build security in, that's what we mean.

We can avoid the "dirty water from clean pipes" landmine by making sure that some of the processes we identify use the best badness-ometers available, things like: security requirements analysis, abuse case development, architectural risk analysis, code review, risk-based security testing, penetration testing, and security operations.

Good security processes leverage common artifacts (for example, McGraw's software security touchpoints[4] are applied to software development artifacts that are usually built anyway during software development…yeah, we know). Many, but not all, software development approaches create the kinds of artifacts we can use to do this.

The BSIMM[30] is a commercial framework that helps organizations measure and improve their software security initiatives by benchmarking their practices against those of other firms. It is based on real-world data and focuses on what successful organizations actually do to secure their software, rather than prescribing specific actions.

BSIMM measurements are, in this sense, second-order measurements because they purport to tell you something about software security without directly measuring the software itself. When McGraw invented the BSIMM with Brian Chess and Sammy Migues in 2009, the Achilles' Heel of the entire measurement system was always gathering clean data. (In fact, though BSIMM16 is the latest published version, post BSIMM9 we no longer have direct confidence or first-hand experience with the dataset.)

All that aside, the point of the BSIMM is to create an objective measuring stick that everyone can use to compare belly buttons. As long as the lines on the measuring stick are the same for everybody, and everybody measures things the same way, this approach works. Look no further than the thermometer to see why.

In final analysis, the BSIMM lesson is not directly transferable to the AI measurement situation…yet. Second-order measurement of assurance processes works for software security because we can collectively identify and measure a well-understood set of 12 practices (each with associated activities)—e.g., training, security features and design, architecture analysis, code review, penetration testing—whose relationship to security outcomes has been established over decades of practice. For AI, we have yet to identify the right practices. We haven't gained enough experience to understand which internal processes, applied to which ML artifacts, reliably produce more secure systems.

At BIML, we believe that before we can build an AI BSIMM, we need a clearer picture of what is actually happening inside these models—which mechanisms produce which behaviors, and where security-relevant signals live in the architecture. To be clear, internal observability is not the security meter Holy Grail itself. Rather, it is a necessary precondition for eventually getting to one.

One day we hope that AI develops something like the BSIMM to measure security. This may well involve a better understanding of the roles that intention and theory of mind play in Agentic AI systems, because proper risk management is often tied up with intention.

Managing Risk in AI: Using ML like a grown up

By this point, we've come to a couple of important realizations: 1) benchmarks for AI are not all they are cracked up to be, and 2) we don't know how to measure security beyond eradicating known badness or adopting second-order measurement regimes anyway. That means even though we want it, we don't get to have a security meter for AI (or a security benchmark for that matter).

The one thing we can get is an idea of how the MLsec tools space is likely to develop. We can predict the (AI) future by understanding the (software security) past. It will likely go something like this:

- First, we will set up and deploy "AI Red Teams" who will carry out bespoke penetration testing focused on prompt injection
- We will generally treat ML system components as black boxes that must be either controlled and monitored or just plain disallowed (we already see many successful "AI Firewall" approaches)
- We will next see Agentic AI controls that boil down to adaptations of human user controls to the "active AI" agent situation (mapping authentication and authorization systems to particular AI Agents)
- Only then we will get inside the AI black box by looking at how ML systems behave internally, moving toward some notions of intention

This reminds us of an important point we need to emphasize. Security is not a thing. For example, security is not magic crypto fairy dust that we can sprinkle over our software to make it secure. If security were a thing, it would be quite possible to just weigh it like some sort of ingredient and make sure we put enough of it into the batter. Instead, security is a non-functional property of a system (in just the same way that dependability, usability, reliability, maintainability, and scalability are properties of a system).

Spitballing AI Security Assurance Touchpoints

Software security matured when practitioners stopped asking "is this software secure?" and started asking "what specific activities, applied at what points in the development process, reliably reduce risk?" The result was a set of touchpoints—architecture analysis, code review, security testing, penetration testing—each targeting a different artifact and a different moment in the development pipeline. Machine learning security is beginning to trace a similar path. Architectural risk analysis of ML systems is emerging as an analog to threat modeling. Model and training code auditing (examining provenance, training data, and development safeguards) loosely resembles the supply chain and configuration management activities that software security practitioners know from deployment. But it is on the dynamic analysis side where the most interesting early work is appearing. In particular, adversarial testing, red-teaming, and agentic stress testing occupy the space that penetration testing holds in software security. Early whitebox approaches—probing internal representations rather than observing external behavior—echo some of the promise of source-level analysis. Papernot and McDaniel's Deep KNN work presaged this direction as far back as 2018, using layerwise nearest-neighbor analysis to expose what a model actually "knows" at each level of representation.[31]

The data dimension complicates our analogy in ways that have no clean software parallel. In software security, the primary artifacts under analysis are largely static—source code and binaries (though software can certainly be dynamically tested at runtime). By contrast, in ML, the training data is itself a primary attack surface: poisoned, contaminated, or simply unexamined data propagates risk in ways that no amount of post-training analysis can fully remediate. A mature AI security touchpoint framework will need an explicit data assurance track (covering at the very least, provenance, contamination detection, and representational integrity) that has no direct predecessor in the software security tradition. Those activities don't yet exist in any systematic form. That is some of what needs to be built.

Opening the Box: Assurance measurements and AI control

The black box nature of many of today's LLM foundation models has a direct impact on measurement (and assurance). Simply put, you can't measure or assess what you can't see. BIML's work on LLM risk makes this problem clear by identifying four central ML components that, by design, are put inside a black box by foundation model vendors.[32]

Early evidence gathered by Starseer (one of a handful of whitebox-focused startups) suggests that opening the ML black box can yield security-relevant insights that benchmarks applied only externally cannot. The basic notion is to look inside of the neural network model at node activation patterns and circuits using a purpose-built "interpretability engine." In a series of unpublished experiments, Starseer's work demonstrates that layerwise activation analysis of transformer models can identify sparse, non-contiguous subsets of internal layers whose activation patterns alone are sufficient to classify jailbreak attempts and prompt injection attacks.

Early testing reveals that this sparse-layer configuration matches or exceeds the classification performance of (external) safety classifiers that are explicitly fine-tuned for the task.[33] Put simply, a small, precise window into a model's internal state can outperform explicit fine-tuning for some types of security classification.

Understanding the underlying whitebox technology unlocks defensive paradigms that benchmarks and black box approaches simply cannot.

The same whitebox internal observability approach extends to controlling model optimization. Starseer's activation divergence analysis identifies which layers inside a network are most responsive to specific behavioral changes, enabling targeted LoRA fine-tuning that retains 97% of full-rank accuracy while training 84% fewer parameters.[34]

This kind of work matters for security because fine-tuning is one of the primary activities through which model behavior is altered—intentionally during alignment, maliciously during attack, or otherwise—and research has shown that most fine-tuning techniques end up altering or removing safety and alignment training.

Architectural risk analysis and threat modeling can both gain accuracy and predictive power when they are enhanced with whitebox insights. A view into which layers carry which behavioral signals provides an opportunity to get past simple questions like "how often does this model pass a battery of tests," toward questions like "where in this model is the relevant behavior engendered, and have alignment mechanisms been tampered with."

This kind of visibility also opens the door to retaining inbuilt safety protections that most fine-tuning techniques would otherwise alter or remove.

To be clear, none of this constitutes a security meter. A skeptic might fairly ask whether layerwise activation classification is simply a more granular badness-ometer—one that finds badness deeper in the stack rather than at the surface. That objection has force. What distinguishes this line of work is not that it measures security directly, but that it begins to expose the internal mechanisms from which security properties emerge. That is a precondition for any future security meter, in the same way that understanding metallurgy was a precondition for marine insurance.

So where does this leave a practitioner making real decisions today? Three things seem clear. First, only use benchmarks for what they are actually good for—comparing models against each other on defined tasks—while refusing to interpret scores as security meters. Second, invest in process: identifying which assurance activities applied to which ML artifacts reliably produce more secure systems is foundational, necessary work that we need to do now. Third, treat internal observability as a research priority. We think that organizations that develop rigorous whitebox analysis methods today will be better positioned to make credible security assurance claims tomorrow. Of course, none of this gives us a security meter for AI, but it gives us a direction.

Measuring AI: From HOW to WHAT to WHY

Simple language may be able to help us think differently about the AI measurement pickle we find ourselves in.

At BIML, we talk about normal computer programs as HOW machines. To put a fine point on it, when you know exactly HOW to accomplish a computable task, you can describe the machine programmatically as a series of deterministic algorithmic steps (e.g., a computer program). We emphasize the word *exactly* here, because computers (as kinds of formal systems) usually require great precision when it comes to programming and are notoriously unforgiving of error.

We may not always be able to describe a HOW machine sufficiently to implement it as a program. In fact, programming has always been a challenge, even for very smart people. That's one of the reasons that machine learning is useful. In cases where we have a (large) set of examples of WHAT we want to accomplish, but we're still not sure HOW to write a program, we can use machine learning.

Imagine we have a huge pile of WHAT that we want to model. We can use machine learning to build a WHAT machine out of a set of training data (that huge pile of WHAT again). The resulting machine is an imperfect (but hopefully useful) model of the WHAT pile. This approach often results in useful WHAT machines that we can consult to solve our WHAT-based problems. But even when they work, we still don't necessarily know HOW they work at the right level of abstraction.

Of course there is a catch—if your WHAT data are poorly chosen, you may reach incorrect conclusions with your ML approach. You should always make sure your methods are a good match for your data and your data are properly vetted and monitored. Anyway, you can imagine the trouble some people go to in order to gather up a nice juicy pile of WHAT.

Let's make this slightly more concrete with an example. We can imagine the idea of writing a program to process a scene made up of 256x256 pixels in a square in order to identify common objects. But when we get down to creating a working HOW machine, this gets much trickier than anticipated because there are too many ways to slice and dice such a picture to write them all down. How many possible images might there be? How do we account for scaling and rotation of common objects we want to recognize, and so on. This generalized object recognition task is ready made for machine learning. The ImageNet dataset (a large WHAT pile all nicely tagged and bagged) was created to develop WHAT machines that could accomplish simple image recognition.

Now that we have some idea about HOW machines and WHAT machines, let's think about security.

It's pretty clear that security in ML has a lot to do with the WHAT pile. Where did the WHAT come from? How clean is the WHAT? Are there Trojans in the WHAT pile? How do we know the WHAT pile describes the thing we want to compute? How much of the WHAT is wrong? Is the WHAT pile biased? Can we protect the WHAT pile? Is the WHAT pile even ours? Simply put, we must control the WHAT.

Of course when you push on these seemingly simple questions, tricky aspects are exposed. Like, who gets to decide what is good and what is bad in a WHAT pile? Who actually owns which parts of a WHAT pile? If we decided to clean all of the bad stuff out of the WHAT pile, how would we do that? (Can we likewise clean all of the micro-plastic out of the ocean?) As you can see, these questions often end up being social and political in nature. Deciding some data are good or bad is a political decision.

We want to emphasize that the WHAT piles behind today's LLM foundation models came directly from society (as captured in various human artifacts). To the extent that the WHAT pile reflects society (in all its chiaroscuro glory), the WHAT machine will embody it in a lossy simulation in order to make predictions.

The machine *is* the data, which is precisely why BIML concerns itself with things like pollution, recursion, and model collapse.

Agentic AI is built as a combination of HOW machines (Agentic harnesses like OpenClaw[35]) and WHAT machines (the LLMs called on to make various underspecified decisions). This stretches both the old kind of computer security (which counts on knowing exactly how a HOW machine is going to behave) and the new computer security (focused on WHAT piles, their curation and management, and stochastic WHAT machine behavior). One path forward is to focus on WHY.

Understanding WHY when it comes to a HOW machine is easy—just look at the program. On the other hand, understanding WHY a WHAT machine does what it does is trickier. Do we ascribe WHAT machines intention and treat them like people? As our systems come to embody more WHAT machines, we will need to start thinking through the WHY.

Our goal at BIML is to peer as far into the MLsec future as possible (in this case looking around in the dark for a security meter). Somewhat ironically, we can do that by remembering the past. We feel like we are reliving an early software security déjà vu. Though that can be discouraging, it also gives us an important super-power. In a Gibson-esque move, we can predict the future.[36]

Today is early days in MLsec. Just like everyone was talking about buffer overflows in 1998 and penetration testing was an absolute wild west blast, everyone is talking about prompt injection in 2026 and AI red teams are all the rage. (And even something as basic as prompt injection is still evolving fast.[37]) That means things in security are likely to shift to monitoring, intrusion detection, and sandboxing just as they did in the early 2000s, first by treating an LLM as a black box whose inputs and outputs need strict oversight. This shift is already well underway. How many AI firewall products are on the market even today? Next, we will realize that we have to get inside the black box and develop a set of assurance practices that are the moral equivalent of code review and architectural analysis. We don't yet know what that means, but work like Anthropic's transformer circuits research is breaking new ground.

Finally we'll invent and flesh out a set of AI assurance practices (which incidentally will include all of the activities above) and start to gather data for an AI BSIMM. In security, some kind of risk transfer (or risk sign-off) is always required. Being able to properly manage AI security risk is an important objective.

Meanwhile, the agentic AI tidal wave is about to hit at "AI-enhanced internet speed." We don't have a security meter for AI yet, but we know what its second-order shape may be, and we know how the story is likely to go since we've lived it before in software security. Strap in, it's accelerando time.

Acknowledgements

BIML would like to thank Brian Chess, Dan Geer, Ron Gula, Patrick McDaniel, Bill Pugh, Tim Shultz, and Jeff Voas for helpful comments and corrections. We would also like to thank our benefactors: Gula Tech Adventures, Anonymous, Ballistic Ventures, Anonymous, George Ohrstrom III, Neil Daswani, Anonymous, Anonymous, and Richard Danzig.

References

See the Berryville Institute of Machine Learning Annotated Bibliography for more commentary and references.

[1] H. Petroski, *To Engineer Is Human: The Role of Failure in Successful Design*. New York, NY: Knopf Doubleday Publishing Group, 1985.

[2] D. Bhusal, M. T. Alam, L. Le Nguyen, A. Mahara, Z. Lightcap, R. Frazier, R. Fieblinger, G. Long Torales, B. A. Blakely, and N. Rastogi, "SECURE: Benchmarking Large Language Models for Cybersecurity," arXiv:2405.20441, 2024.

[3] S. Shahriar and R. Dara, "Priv-IQ: A Benchmark and Comparative Evaluation of Large Multimodal Models on Privacy Competencies," *AI*, vol. 6, no. 2, p. 29, 2025. doi: 10.3390/ai6020029.

[4] G. McGraw, *Software Security: Building Security In*. Upper Saddle River, NJ: Addison-Wesley, 2006.

[5] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2009.

[6] A. Krizhevsky, I. Sutskever, and G. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," *Advances in Neural Information Processing Systems*, vol. 25, 2012.

[7] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention Is All You Need," arXiv:1706.03762, 2017.

[8] A. Power, Y. Burda, H. Edwards, I. Babuschkin, and V. Misra, "Grokking: Generalization Beyond Overfitting on Small Algorithmic Datasets," arXiv:2201.02177, 2022.

[9] J. Hoffmann, S. Borgeaud, A. Mensch, E. Buchatskaya, T. Cai, E. Rutherford, D. de Las Casas, L. A. Hendricks, J. Welbl, A. Clark, T. Hennigan, E. Noland, K. Millican, G. van den Driessche, B. Damoc, A. Guy, S. Osindero, K. Simonyan, E. Elsen, J. W. Rae, O. Vinyals, and L. Sifre, "Training Compute-Optimal Large Language Models," arXiv:2203.15556, 2022.

[10] D. Hendrycks, C. Burns, S. Basart, A. Zou, M. Mazeika, D. Song, and J. Steinhardt, "Measuring Massive Multitask Language Understanding," in *Proc. Int. Conf. Learning Representations (ICLR)*, 2021.

[11] M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. de Oliveira Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman, A. Ray, R. Puri, G. Krueger, M. Petrov, H. Khlaaf, G. Sastry, P. Mishkin, B. Chan, S. Gray, N. Ryder, M. Pavlov, A. Power, L. Kaiser, M. Bavarian, C. Winter, P. Tillet, F. P. Such, D. Cummings, M. Plappert, F. Chantzis, E. Barnes, A. Herbert-Voss, W. H. Guss, A. Nichol, A. Paino, N. Tezak, J. Tang, I. Babuschkin, S. Balaji, S. Jain, W. Saunders, C. Hesse, A. N. Carr, J. Leike, J. Achiam, V. Misra, E. Morikawa, A. Radford, M. Knight, M. Brundage, M. Murati, K. Mayer, P. Welinder, B. McGrew, D. Amodei, S. McCandlish, I. Sutskever, and W. Zaremba, "Evaluating Large Language Models Trained on Code," arXiv:2107.03374, 2021.

[12] P. Clark, I. Cowhey, O. Etzioni, T. Khot, A. Sabharwal, C. Schoenick, and O. Tafjord, "Think You Have Solved

Question Answering? Try ARC, the AI2 Reasoning Challenge," arXiv:1803.05457, 2018.

[13] K. Cobbe, V. Kosaraju, M. Bavarian, M. Chen, H. Jun, L. Kaiser, M. Plappert, J. Tworek, J. Hilton, R. Nakano, C. Hesse, and J. Schulman, "Training Verifiers to Solve Math Word Problems," arXiv:2110.14168, 2021.

[14] R. Zellers, A. Holtzman, Y. Bisk, A. Farhadi, and Y. Choi, "HellaSwag: Can a Machine Really Finish Your Sentence?" in *Proc. 57th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2019.

[15] Open LLM Leaderboard, Hugging Face. [Online]. Available: https://huggingface.co/spaces/open-llm-leaderboard/open_llm_leaderboard#/. [Accessed: Feb. 7, 2026].

[16] Scale AI Leaderboard. [Online]. Available: https://scale.com/leaderboard. [Accessed: Feb. 7, 2026].

[17] Vellum LLM Leaderboard. [Online]. Available: https://www.vellum.ai/llm-leaderboard. [Accessed: Feb. 7, 2026].

[18] Arena AI. [Online]. Available: https://arena.ai. [Accessed: Feb. 7, 2026].

[19] M. Mitchell, "The Metaphors of Artificial Intelligence," *Science*, vol. 386, no. 6723, p. eadt6140, 2024. doi: 10.1126/science.eadt6140.

[20] M. Mitchell, "On Evaluating Cognitive Capabilities in Machines (and Other 'Alien' Intelligences)," AI: A Guide for Thinking Humans, Substack, Jan. 14, 2026. [Online]. Available: https://aiguide.substack.com/p/on-evaluating-cognitive-capabilities.

[21] M. Sanz-Gómez, V. Mayoral-Vilches, F. Balassone, L. J. Navarrete-Lozano, C. R. J. Veas Chavez, and M. del Mundo de Torres, "Cybersecurity AI Benchmark (CAIBench): A Meta-Benchmark for Evaluating Cybersecurity AI Agents," arXiv:2510.24317, 2025.

[22] Y. Wu, J. Smith, A. Patel, L. Chen, M. Gómez, H. Thompson, and V. Rao, "ExCyTIn-Bench: Evaluating LLM Agents on Cyber Threat Investigation," arXiv:2507.14201, 2025.

[23] H. Wang, Q. Mang, A. Cheung, K. Sen, and D. Song, "How We Broke Top AI Agent Benchmarks: And What Comes Next," Center for Responsible, Decentralized Intelligence, UC Berkeley, Apr. 2026. [Online]. Available: https://rdi.berkeley.edu/blog/trustworthy-benchmarks-cont/.

[24] S. Abdulaziz, G. D'Amicantonio, and E. Bondarev, "Evaluation of Human Visual Privacy Protection: A Three-Dimensional Framework and Benchmark Dataset," arXiv:2507.13981, 2025.

[25] H. Li, W. Hu, H. Jing, Y. Chen, Q. Hu, S. Han, T. Chu, P. Hu, and Y. Song, "PrivaCI-Bench: Evaluating Privacy with Contextual Integrity and Legal Compliance," arXiv:2502.17041, 2025.

[26] D. Pau, C. Bachot, C. Monteil, L. Vinet, M. Boucher, N. Sella et al., "Comparison of Anonymization Techniques Regarding Statistical Reproducibility," *PLOS Digital Health*, vol. 4, no. 2, p. e0000735, 2025. doi: 10.1371/journal.pdig.0000735.

[27] G. McGraw and G. Hoglund, *Exploiting Software: How to Break Code*. Boston, MA: Addison-Wesley, 2004.

[28] D. Farmer and W. Venema, "Improving the Security of Your Site by Breaking Into It." [Online]. Available: https://www.academia.edu/83242811/Improving_the_security_of_your_site_by_breaking_into_it. [Accessed: Apr. 15, 2026].

[29] "Beyond the Badness-ometer," Dr. Dobb's, Jun. 30, 2006. [Online]. Available: https://web.archive.org/

web/20060710082904/http:/www.ddj.com/dept/architect/189500001. [Accessed: Apr. 15, 2026].

[30] G. McGraw, "BSIMM." [Online]. Available: https://www.garymcgraw.com/technology/bsimm/. [Accessed: Feb. 7, 2026].

[31] N. Papernot and P. McDaniel, "Deep k-Nearest Neighbors: Towards Confident, Interpretable and Robust Deep Learning," arXiv:1803.04765, 2018.

[32] G. McGraw, H. Figueroa, R. Bonett, and K. McMahon, "An Architectural Risk Analysis of Large Language Models: Applied Machine Learning Security," Berryville Institute of Machine Learning (BIML), Creative Commons, Jan. 24, 2024. [Online]. Available: https://berryvilleiml.com/results/BIML-LLM24.pdf.

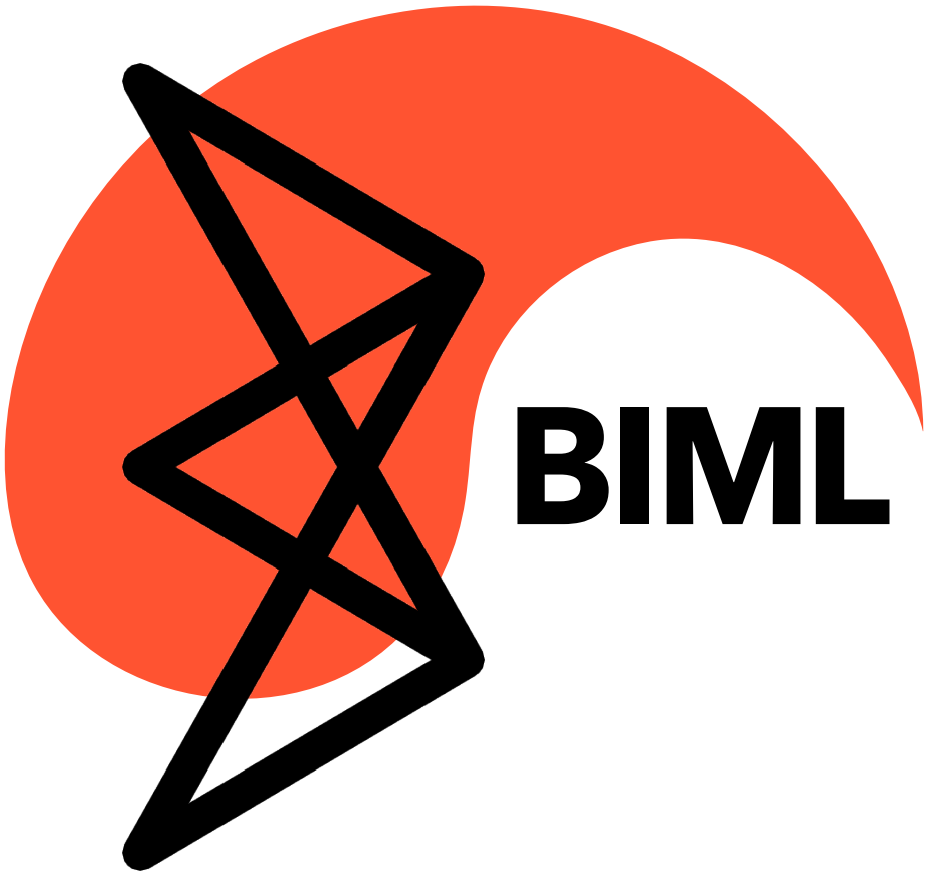
[33] Starseer, personal communication, Apr. 15, 2026. [Online]. Available: https://starseer.ai/home.

[34] A. Arditi, O. Obeso, A. Syed, D. Paleka, N. Panickssery, W. Gurnee, and N. Nanda, "Refusal in Language Models Is Mediated by a Single Direction," arXiv:2406.11717, 2024.

[35] P. Steinberger, "OpenClaw," GitHub, Nov. 2025. [Online]. Available: https://github.com/openclaw/openclaw. [Accessed: Apr. 15, 2026].

[36] W. Gibson, "The Science in Science Fiction," *Talk of the Nation*, NPR, Nov. 30, 1999.

[37] OpenAI, "Designing Agents to Resist Prompt Injection," Mar. 11, 2026. [Online]. Available: https://openai.com/index/designing-agents-to-resist-prompt-injection/.


BIML