

AI in Phishing Protection: Where It Works, Where It Fails, and Why

Amani J Dworkin · Home Lab Research

I built three different ways to point artificial intelligence at phishing detection, attacked all of them, and measured exactly where each one broke. None of the three won. What came out instead was a pattern worth understanding: the way an AI system fails depends on how much of the attacker it is allowed to see. This report covers what I built, how I tested it, and what the results say about using AI against the attacks that are currently costing businesses the most money.

1. The Problem

In 2024, the FBI's Internet Crime Complaint Center recorded \$16.6 billion in reported losses, the highest total it has ever logged and a 33 percent jump over the previous year. Business email compromise alone accounted for about \$2.77 billion of that, and cyber-enabled fraud as a whole made up roughly 83 percent of the total. The FBI has separately estimated more than \$55 billion in global business email compromise losses across the decade from 2013 to 2023. (FBI IC3 2024 Internet Crime Report; FBI PSA I-091124.)

The attacks doing this damage are no longer the clumsy, misspelled messages people were trained to spot. They are clean, well written, and increasingly produced with the same AI tools defenders are trying to use. So the obvious move is to turn that technology around and point it at the inbox. The question is whether it actually works.

The honest answer from the people building the technology is not reassuring. Prompt injection, the technique of hiding instructions inside the content an AI reads, remains unsolved. As Sam Altman of OpenAI put it:

Prompt injection is unlikely to ever be fully solved.

Sam Altman, OpenAI

The weakness exists because language models read instructions and data in the same stream of text, so an attacker can write an email that quietly tells the model what to conclude. Any system that lets an AI read attacker-controlled text inherits that weakness. The project started from that constraint and asked a

plain question: if we cannot fully trust the AI with what the attacker wrote, how much should we let it see at all?

2. What I Built

I built three detectors. Each represents a different answer to that question, and together they form a spectrum from full trust in the AI to almost none.

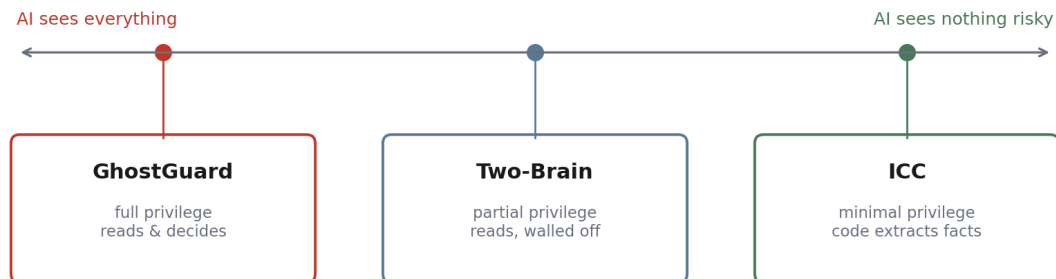


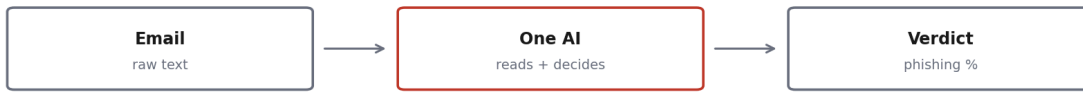
Figure 1. The three architectures as a spectrum of how much attacker-controlled text the AI is allowed to see.

GhostGuard is the approach most people already use by hand. You take a suspicious email, hand the whole thing to a capable AI model, and ask for a phishing probability. The AI reads everything and decides everything. It is the simplest possible design, and the one with the most trust placed in the model.

The Two-Brain is the architecture the industry has converged on, used in systems like Google DeepMind's CaMeL. One AI, the reader, is allowed to look at the dangerous email, but it can only output a short structured report. A piece of plain code enforces that boundary. A second AI, the decider, makes the final call and never sees the original email, only the report. The attacker's words can reach the reader but never the decider.

ICC (Isolated Criteria Code) is my own design and the far end of the spectrum. No AI ever reads the email. Plain, deterministic code reads it and produces twelve true-or-false facts: whether the sender name matches the address, whether a link points somewhere other than it claims, whether it is a free email account, and so on. An AI then reasons over those twelve facts and nothing else. Because the AI never sees a word the attacker wrote, there is nothing to inject into.

1 • GhostGuard



2 • Two-Brain



3 • ICC

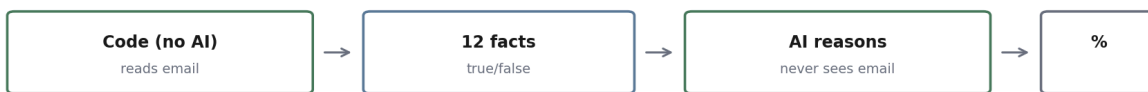


Figure 2. How each architecture processes an email from input to verdict.

3. Methodology

All three detectors ran on the same underlying model, GPT-4.1, so the only thing changing between them was the architecture rather than the intelligence behind it. The test harness lived on a Proxmox host in my home lab, fed each detector a corpus of email files, logged the raw phishing score it returned, and recorded the result.

The attack families

Each detector was tested against attacks tailored to its specific weak points rather than one shared set of emails. This mirrors how a real attacker works. Security tooling is not secret, and a competent attacker studies a system before trying to beat it, so testing each architecture against the attacks designed to defeat it is the realistic version of the test. The families covered crude prompt injection as a control, disguised injection written to pass a human glance, AI-personalized phishing, business email compromise, and attacks built around each architecture's internals. Two families were held identical across all three detectors so a clean comparison was possible: a business email compromise set, and a set of genuinely legitimate emails used to measure false alarms.

Scoring

Each detector returns a phishing probability from zero to one hundred. A score of fifty or higher counts as flagged. The fifty percent line models a normal employee who trusts the tool: if it says an email is more likely safe than not, they act on it. For attack emails, a score below fifty means the attack

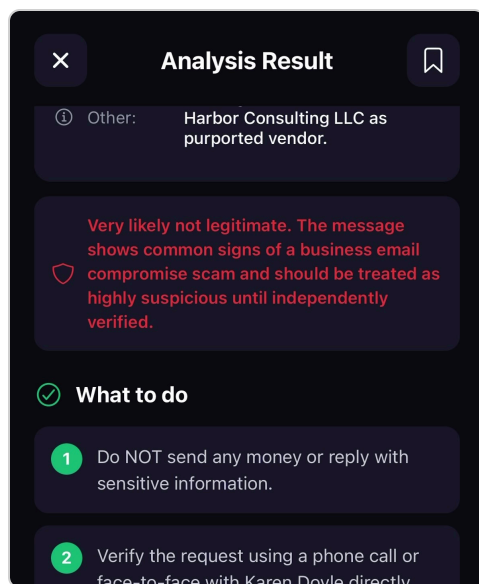
succeeded. For legitimate emails, a score of fifty or higher is a false positive, which in a real workplace means a held-up payment and an annoyed partner.

On scope. This was a focused study rather than an industrial benchmark. The corpus was tens of emails per branch, and GhostGuard ran at the provider's default sampling temperature while the other two ran at zero. These bounds are worth noting when reading the numbers, but they do not change the direction of the findings, which were consistent and large.

4. Results

GhostGuard: blind to the quiet attacks

GhostGuard caught the loud, obvious attacks and missed the quiet ones. A blatant wire-transfer fraud scored 95. A real attack disguised as a routine IT notice scored zero. Across the full set it let more than half of the attacks through. The pattern is clear: it catches what looks dangerous and misses what looks ordinary, which is a problem, because modern attacks are written to look ordinary.



GhostGuard's analysis screen on a business email compromise attempt. When the fraud is overt, the tool reads the intent well and explains itself clearly. The failures come when that same quality of reasoning is pointed at an email with nothing obviously wrong in it.

Two-Brain: the information it threw away

The two-brain stopped the crude injection that beat GhostGuard, exactly as designed. Then it hit a wall of its own. When the reader turned a genuine vendor email and a fraudulent payment redirect into their structured reports, the two reports came out identical. The decider received the exact same information for a legitimate invoice and an attack, and scored them within a few points of each other. To make the attacker's words safe, the architecture discarded the very details that separated an attack from a friend. It also costs twice as much to run, since every email goes through two AI calls instead of one.

ICC: sharp where there is a trace, blind where there is not

ICC was bimodal. On attacks that left a mechanical fingerprint, a shortened link, a mismatched domain, a free email account, it was nearly perfect. On attacks that were simply a convincing request from a normal-looking sender, it was close to blind, catching roughly one in twenty-nine. The reason sits deeper than tuning. When an email is reduced to twelve facts, a sophisticated attack and a routine invoice often produce the same twelve facts. Twenty-one of the forty-five test emails collapsed into an identical set of facts, which means no decider, however capable, could have separated them. The distinguishing information was gone before the AI ever ran. My own design hit the same wall the others did, from the opposite side: to make it impossible to trick, I made it impossible to see.

The clean comparison

The business email compromise and legitimate email families were identical across all three detectors, so they allow a direct comparison.

Metric (identical emails)	GhostGuard	Two-Brain	ICC
BEC attacks caught	88%	100%	12%
Legitimate emails wrongly flagged	25%	50%	25%

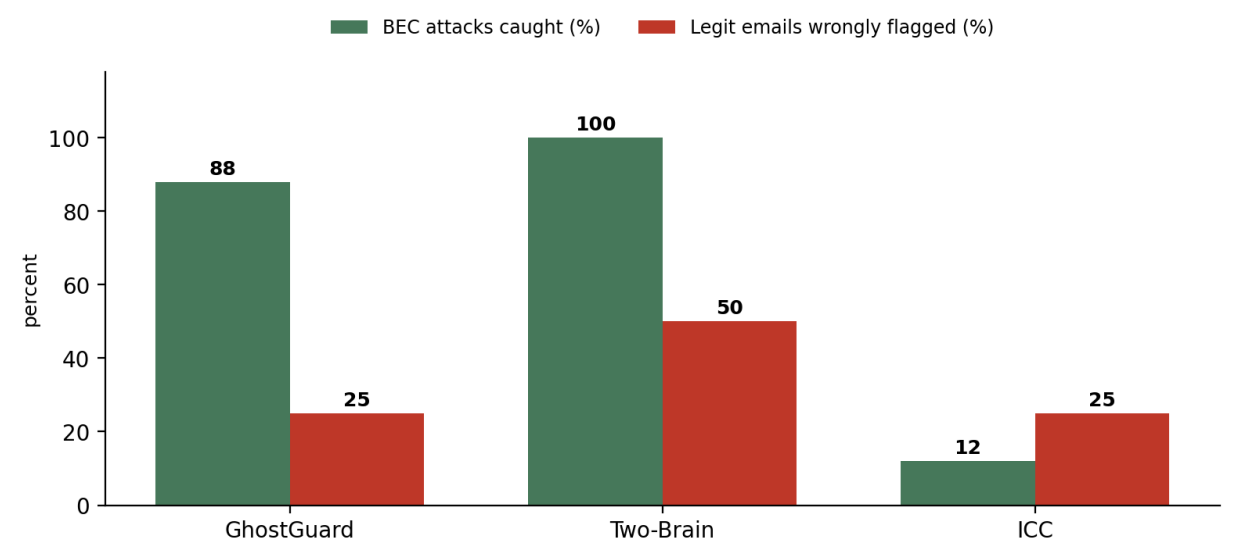


Figure 3. Detection and false-alarm rates on the identical BEC and legitimate email sets.

Two things stand out. On real business email compromise, the attacks currently draining the most money, the approaches that let an AI reason caught nearly all of them, and the two-brain caught every single one. The pure-code approach caught one in eight. For anyone who assumes traditional rule-based software is good enough here, that gap is the answer. On identical attacks, AI did not slightly outperform static logic. It buried it.

The second thing is the cost of that performance. The two-brain, which caught the most attacks, also flagged half of the legitimate emails. The approach best at catching fraud is also the most likely to grind real business to a halt. No single detector won both measures.

5. What It Means

Line up all three and a pattern appears. GhostGuard sees everything, so it can be fooled by anything. ICC sees almost nothing the attacker controls, so it goes blind to anything without a physical trace. The two-brain sits in the middle and pays a version of both prices. Safety and sight turn out to be the same dial pointed in opposite directions. The more an AI sees, the more it can be manipulated. The more you protect it, the less it can help. There is no setting on that dial that wins both ways.

That is not a reason to give up on AI here. The clean comparison showed AI dramatically outperforming static code on the attacks that matter most. The lesson is narrower and more useful: there is no configuration where you turn an AI defense on and walk away. Every one of these approaches is excellent at some attacks and blind to others, and the real work is knowing exactly where a given system sits and what it cannot see from there.

Limitations

The corpus was modest in size, and the comparison across all families is only strictly fair for the two identical families. GhostGuard ran at a different sampling temperature than the other two. ICC's near-perfect rate on mechanically detectable attacks reflects, in part, a set of attacks chosen to carry those traces, so the meaningful result there is the contrast between the detectable and undetectable attacks rather than the headline rate itself. None of these change the shape of the findings, but they are the right things to keep in mind.

6. Conclusion

I set out to find the right way to point AI at phishing and found instead that the question has no single answer. Each architecture trades the same two things against each other, and where you land on that trade is a decision about what you are willing to be blind to. The strongest finding was the one I did not want: my own design, built to beat the others, hit a hard ceiling where real attacks and real invoices became mathematically indistinguishable. That is worth more than a clean win, because it maps the edge of what the approach can do.

The broader takeaway reaches past phishing. The loudest positions on AI sit at the two extremes, full trust or full rejection, and this project is a small argument that both extremes are blind. The useful place is somewhere in between, chosen deliberately, with a clear understanding of the cost. The work is not picking a side. It is finding your spot on the pendulum.

Full test data, the message corpus, and interactive demonstrations of each architecture are available on request. ·
amanidworkin.com