ЙОВДІЙ ГРИГОРІЙ

Вибрані тексти · англомовний B2B-контент

098 537-49-06 · apshanuivd@gmail.com · github.com/Gregory-Ivd

Про цю підбірку

Чотири тексти, написані для digital-агенції Livepage у 2025 році. Кожен — outreach-матеріал для клієнта агенції: технологічна ніша, англійська мова, жорстке ТЗ і SEO-вимоги, ключове слово та анкор на сторінку клієнта.

Усі чотири прийняті агенцією. Опублікованих посилань я не наводжу свідомо: outreach-матеріали виходять на сторонніх майданчиках під їхнім брендом, часто з переписаним заголовком і без імені автора. Тому тут — авторські версії.

Тексти вичитані перед укладанням підбірки. Зміст, структура й аргументація — авторські, без змін.

Що в підбірці

N-iX · кібербезпека — The Self-Driving Hacker: How AI Is Rewriting Cybercrime

N-iX · генеративний AI — Why Your AI Feature Costs More Than Your Entire Backend

PFLB · навантажувальне тестування — The Release That Crashed at Midnight

SmartyAds · programmatic — When Your DSP Knows More About the User Than Your Product

N-iX · Cybersecurity · B2B tech

Ключове слово: cybersecurity services provider

Показує: технічна конкретика замість загальних слів; анкор, вбудований у сенс

The Self-Driving Hacker: How AI Is Rewriting Cybercrime

Cyberattacks are no longer a matter of intent alone. They have become self-developing systems in which AI copies defence methods and adapts them into highly individualised attacks. Most companies do not even suspect that while the SOC panel shows green, an algorithm is already working in the shadows, learning from their logs.

A cybersecurity services provider can help build defences around behaviour rather than incidents. The question has changed: it is no longer "are you being attacked," but "is someone learning from your data right now?" Let's talk — for example, about how to protect yourself from those who are already inside, operating at computing speeds your firewalls have not yet learned to think with.

When the Monitor Lies: An Attack Without a Keyboard

There is a moment when the system is still alive — it answers requests, CPU is within range, latency is within the SLA, error rates are not jumping — and yet it is already compromised. The dashboard lulls you with green graphs: normal TPS, stable API response, scheduled nightly builds running on time. You read all this as the health of your infrastructure: green equals OK.

Under the appearance of a working system, however, a leak begins. Usually with a quiet beacon: rare outgoing connections to unfamiliar domains, invalid tokens in 0.01% of requests, new user agents in the logs, unusual parent-to-child process chains such as python spawned by sshd, or small, regular spikes in outgoing traffic at night. None of this raises an alarm on its own. Together they form a consistent pattern.

Do not expect the SOC panel to figure it out by itself. Ask an engineer to show you three things:

- a list of unusual egress domains;
- processes with non-standard parent-to-child relationships over the last 24 hours;
- your MTTD — and ask why it is still above five minutes.

The attack is already underway. Without a keyboard, and without a human.

An autonomous agent has scanned your stack, tested updates, recorded how long the SOC took between anomaly and first response, and started generating new payloads faster than your standard alert can fire. It does not wait for orders. It decides for itself where the weak link is.

The Rise of Autonomous Offence: When Hacking Became a Service

In the past, a serious attack required brains, time and nerve. Today it requires an API key to an open LLM. The model generates the scenario, combines exploits, probes your firewalls and adapts to a specific infrastructure. No noise. No keyboard. No human. And it does so with unsettling accuracy — faster than you can notice that something has gone wrong.

How an Attacker Calculates Efficiency

Expected value = P(success) × Value(success) – Cost(attack).

Autonomous agents raise P(success) and lower Cost(attack), so expected value rises. The time to repeat an attempt is close to zero, which makes the strategy both repeatable and scalable. That arithmetic, not malice, is why scanning is now indiscriminate.

The hacker is a model, not a person — one that learns from open datasets and from your own logs. SOC teams are trained on behaviour patterns: if X, then Y. Autonomous agents do not repeat themselves. They imitate at every iteration and create new patterns faster than signature-based systems can recognise them.

Technology is neutral. It serves whoever is faster, and that currently includes the attacker.

Mirror War: The Symmetry of Algorithms

There are no villains and heroes in this race. There are two AIs, two streams of computation, two languages of speed.

The attacking model probes your ports and learns from every failure. Your defensive model analyses millions of events and builds a behavioural picture of the threat. They are twins separated by a wall.

That is why the old logic of defence is dying. Security is no longer built on fortresses but on reflexes. You cannot isolate yourself from the system, but you can teach it to respond quickly. This is how AI-based protection works: it reacts not to threats but to deviations. It registers that something is wrong before it escalates into an attack.

As autonomous defence becomes a mirror image of autonomous attack, working with a cybersecurity services provider that understands both sides of the equation stops being optional.

Mean Time to Adapt: A New Security Measure

"Hackers aren't interested in us, we're too small" is a comfortable sentence. It is also obsolete. Attacks no longer choose their victims; they scan everyone. The model does not know who you are and does not care. It is looking for a weak link, and if it finds one, that is luck for the algorithm, not for you.

Modern attacks target your speed rather than your data: the longer you deliberate, the wider the window you leave open.

The standard security measure is the time it takes to detect and respond. Mean Time To Detect covers the gap between the start of an incident and its discovery; Mean Time To Respond covers the gap between discovery and containment. When attacks are automated and scalable, every minute you shave off those two numbers is a minute the attacker loses.

Conclusion

There is a moment between awareness and action — about a second. Sometimes that is enough to survive.

Adaptive security teams treat speed not as a metric but as architecture, embedding real-time learning loops directly into infrastructure.

Cybersecurity has stopped being a question of technology. When everyone has access to the same tools, the only variable that matters is reaction speed — the difference between control and collapse. Speed is not a metric here. It is an instinct, and it is worth designing for.

N-iX · Generative AI · FinOps · B2B tech

Ключове слово: generative AI consulting

Показує: сцена замість вступу, економіка замість захоплення технологією

Why Your AI Feature Costs More Than Your Entire Backend

Three weeks ago you shipped an AI feature in a fintech product: an assistant that translates the legal language of loan agreements into plain speech.

Users are delighted. Requests to the assistant are up 340%. NPS has moved from 32 to 68. The product team is already scoping version two.

Then a message from the CFO arrives with no subject line. Attached is an invoice from the model provider for $47,000. For one month.

First question: how much revenue did it generate?

There is no answer. Nobody calculated it. Nobody knows how to calculate it. The goals were written as "increase retention" and "improve engagement," and the feature did exactly that. At what unit cost, nobody can say.

We are shipping generative AI faster than we can understand its economics. That gap is the real entry point into generative AI consulting — the moment it becomes necessary to count the economics rather than the tokens.

The Paradox of Invisible Costs

In the new world everything looks good: cloud graphs climb, metrics shine, users are happy. Behind every token sits a microtransaction, and behind every generation sit dollars. All of it dissolves into the "miscellaneous" line of the P&L.

There is something absurd about the modern AI economy: you can roll out an entire feature that feels free, right up until the invoice lands. We know how to budget storage, RAM and SLA. We have not learned to budget the cost of inference built directly into the user experience.

The reason is that AI is still treated as a black box of efficiency rather than a managed service with a measurable unit cost. The real expense of a model is the price of not controlling its economics.

The Illusion of Effectiveness

Metrics keep rising. Users stay satisfied. A month later a strange hole appears in the financial report.

One e-commerce team rolled out an AI recommendation service. The good news: retention rose, and so did click-through. The bad news: net profit fell by almost 12%. **Inference cost more than the returning customers were worth.**

The technology was not the problem. The problem is that AI features are measured in customer sentiment and paid for out of your infrastructure budget.

The CTO says: the system is learning. The CFO answers: and it is learning on my account.

A sceptic will say innovation is always expensive at first. True. But engineers optimise latency, not margin, and product managers measure engagement, not return on inference. The result is a feature that behaves less like a growth driver and more like a cost centre wearing the costume of innovation. We got used to asking whether it works and forgot to ask what it costs.

The Anatomy of AI Costs

Treat the economics of your model not as a piece of code but as a living system with its own rhythms, dependencies and need for attention. Without that, degradation is a matter of time.

1. It has a brain — inference, which makes the decisions.
2. It has a body — the infrastructure that keeps it running.
3. It has blood — data, without which the rest is an expensive set of calculations.

Most companies see only the brain. The rest are treated as details that will sort themselves out.

Every team has that one meeting where nobody asks what the API actually costs.

In a complex model the API line is not a line at all, it is an ecosystem of dependencies. A cache failure costs you optimisation. A delay in retraining costs you context. A compliance gap costs you a fine. None of that appears in the inference bill.

Component	Visible cost	Hidden cost	Symptom when ignored
Model API	$ per 1,000 tokens	Drift, caching, latency	Billing anomalies
Storage and logs	Cloud fees	Audit, compliance, security overhead	Unpredictable overhead
Human feedback loop	Annotator payments	Iteration delays, loss of contextual accuracy	Gradual quality degradation

These are the items that decide whether your AI product becomes sustainable or turns into a chain of experiments with a predictable ending. Which is why "let's add a model" in a planning meeting sounds about as considered as "let's add an engine" to a glider. Intriguing. Unlikely to take off.

Every token has a body. Every expense has a metric. Every company reaches the point where it has to learn to see and count them. That point is where building *with* intelligence starts, rather than building *for* it.

When Anti-AI Wins

There are cases where rejecting an AI solution is not a defeat but a sign of competence.

One company tested an LLM assistant for user support. After an audit it found that 60% of requests could be handled by an updated knowledge base and parametric search. No model. No tokens. Tens of thousands of dollars saved.

Sometimes the anti-AI route is not a step backwards but architectural maturity: updated templates, a couple of training sessions for operators, and inference costs drop by 40% with no loss of quality.

Some will object that this is not innovation. It is evolution — the kind that pays salaries.

Conclusion

The true cost of AI may not sit in tokens or GPUs at all, but in the time spent convincing ourselves that progress equals more computation.

AI has become a mirror of organisational maturity. It shows where a company genuinely knows how to count, and where it simply believes the graph.

Three doubts worth keeping:

- What if we are paying not for results, but for the illusion of control?
- What if generative AI consulting is just accounting applied to ideas?
- What if it is not the costs we are afraid of, but seeing them?

If your AI feature costs more than the entire backend, the thing to rebuild may not be the model. It may be the definition of value.

So: version two, or a sustainable architecture?

PFLB · Performance testing · QA · B2B

Ключове слово: performance testing service

Показує: робота з цифрами й джерелами; переклад інженерної теми на мову ради директорів

The Release That Crashed at Midnight: Why Performance Testing Belongs in the Boardroom

00:01. New release. Shiny slides, glossy banners, and the code nobody dared to test at 2 a.m. goes live. The register button is dead. Not slow. Not laggy. Dead.

Slack is on fire, PagerDuty is screaming, investors are calling the CEO. The CTO turns pale and repeats it like a mantra: we have an SLA. That is the moment the real failure shows itself. An SLA proves the server is alive. It does not prove the system can carry real load.

Which brings up the practice that usually gets pushed to the bottom of the list: choosing and implementing a performance testing service. Not as one more tool on a slide, but as the only practice capable of simulating real user behaviour before the market does it for you at one minute past midnight.

SLA vs Reality

In engineering terms, an SLA describes service viability in the current configuration. It confirms the server responds within formal metrics — uptime, latency, error rate. It says nothing about how the service behaves in production under real load, when a hundred thousand students hit "start exam" in the same second, or when a fintech platform receives a million transactions in a single minute.

Everyone in IT knows downtime is expensive. The numbers show how brutal it gets. 84% of companies lose at least $10,000 per downtime incident, and average downtime cost in IT reaches $9,000 per minute. 98% of organisations say an hour of interrupted operations costs them more than $100,000. In e-commerce, up to 32% of customers never come back after a single failure.

This is why load and resilience testing gets brought in early: most so-called cloud failures have nothing to do with AWS or Azure. They are architecture failures. Money is the first punch. Reputation is the knockout. Twitter jokes do not disappear, and Reddit threads stay indexed.

The Cost of Downtime vs the Cost of Testing

Twenty minutes after the release. The team runs on caffeine, Slack keeps pinging, the CTO writes: all green, we're good. The Black Friday promo is live. Landing pages deployed, discounts rolled out across the database, push notifications sent by segment. Everything looks exactly as it looked in the test environment. It will not hold.

Digital customers do not write to support and do not make a scene. They go to a competitor and press "buy" there. The market is not angry with you; it is simply absent. Your dashboards show it as abandoned carts, falling conversion and silent churn.

The financial hit is only the first stone. Then the chain reaction starts.

Reputation. It collapses when, half an hour after release, people are already joking online about your crash instead of your cash. Reddit delivers the verdict: another startup that did not live up to the hype.

Team. Red eyes, third night without sleep. Coffee stopped working a while ago. Commits go straight to production, bugs get patched on the fly. Fatigue turns into frustration, and a few weeks later an engineer posts one word on LinkedIn: burnout.

Investors. Flat tone on the phone, like a banker reading a bad loan file: what was that? A couple of months later one of them quietly exits the round. No drama. Just less money.

Put together: the brand is associated with failure rather than innovation, people leave, investors lose confidence. And none of it started with memes. It started with twenty minutes of downtime that nobody tested in advance.

From Logs to Revenue

This is not an engineering conversation. In the boardroom nobody cares about logs. They care about revenue lines turning red. When a release fails, it is not the code that collapses but the model the entire pitch was built on.

Why Testing Belongs at the Top

Capacity planning has to be signed off at board level. Resilience and scalability testing is not a QA toy; it is part of market entry, and it belongs alongside finance, marketing and go-to-market.

Ask yourself: you are showing the board a user growth chart. Fine. Is there a second line next to it saying the system will hold at peak load? If not, that is not a strategy, that is a hope. Hope at scale — when marketing drives hundreds of thousands of clicks and the system breaks on the first peak — costs millions. One night-time crash turns a launch into a meme and a company into a cautionary slide in somebody else's deck.

The Real Cost of Skipping Tests

Performance testing costs roughly a week of engineering time. One downtime incident costs, on average, a hundred weeks of the same team. Skipping it can cost you an investor walking out of the room. The CFO understands that faster than anyone, and no spreadsheet is required.

The Reality Check Every Board Should Demand

- **Growth curve** — the chart you show investors.
- **Resilience curve** — what the system actually survives at midnight.
- **Failure map** — what breaks first, and how fast you recover.

Teams that live inside large-scale load simulations keep repeating the same sentence in boardrooms: growth metrics mean nothing without resilience metrics. The shift happens when performance testing stops being a checkbox for engineers and becomes a board-level technology decision. The market always tests with reality, and reality does not read your deck.

Conclusion

You checked everything except the only thing that mattered. Reality showed up precisely when there was no plan B and no coffee left.

Three objections you will hear in every boardroom:

- "What if the integration fails?" Better to find out in testing than in the support queue.
- "Testing is expensive." Watching a launch collapse in front of investors is more expensive.
- "Everybody does it this way." Except the ones who win.

So, the blunt question: in your release plan, is performance testing a budgeted step — or faith that the SLA will hold?

SmartyAds · AdTech · programmatic · product analytics

Ключове слово: demand-side platforms

Показує: складна тема через одну впізнавану ситуацію; міст між двома відділами

When Your DSP Knows More About the User Than Your Product

Have you ever caught yourself thinking that an advertising platform knows more about your customer than your own product does?

The marketing manager brings a report where demand-side platforms supply a full behavioural profile: when the person clicked, what content they met on the way, how many times they saw the banner, which impression finally worked. Then you open product analytics. Two fragments: purchase and drop-off. That is all.

The feeling that follows is strange. As if someone keeps a parallel diary of your users and does not show it to you.

What the DSP Sees

A DSP works like a large sensor array. It records not only the click but the context: which app the user was in, how many times the ad was served, how often it converted. It is telemetry — dozens of sensors capturing every micro-step.

Product analytics looks at events inside the product: registration, purchase, retention. The entire journey to your door is reset to zero at the threshold.

The real pain is in the gap between two sentences. Marketing knows that these people respond to a trigger message on the third impression. The product only knows that retention is 20%.

A sceptic will say there is GA, Amplitude, Mixpanel. They show movement inside your funnel, and only inside it. They will not tell you which behavioural signals brought the user to the threshold in the first place.

The result is that you do not have one customer. You have two — an advertising customer according to the DSP and a product customer according to analytics — and no way to confirm they are the same person.

When Data Lives in the DSP and the Product Stays Blind

This scenario repeats across companies. The CTO shows the retention curve: people drop off on day five. The CMO opens the DSP report: but look, a hundred thousand unique contacts a week.

Both are right. Both are speaking different languages.

Technical barrier. DSP data lives in closed ecosystems. APIs are limited, formats are incompatible, updates arrive later than the product needs them. Marketing sees the click, the product sees the purchase, and the space between them stays empty.

Organisational barrier. Marketing does not share raw logs — too noisy, too difficult. The product says it does not need them right now. This continues for months, until retention drops and there is nobody to hold responsible.

Mental barrier. Two coordinate systems. One team believes in reach, the other lives by retention. They argue about which matters more instead of asking what the customer actually remembers.

On demo day it becomes almost physically uncomfortable. The chart with a million impressions sparkles; the repeat-visit line is flat. The investor asks who came back a second time, and the room has no answer. That is not a bug, it is a diagnosis: money goes into acquisition, retention leaks.

What actually helps is unfashionable: interviews, surveys, cohort analysis, even in a spreadsheet. They capture what no demand-side platform can supply — memory, motive, the reason for coming back. That is often enough to save a product from living with two parallel truths: there was contact, and the customer is gone.

The Bridge Between DSP and CDP

This is where the engineering starts.

Feed DSP data into a Customer Data Platform, and from there into product analytics, and the most frustrating gap closes: the one between acquisition and retention. A CDP assembles user profiles from advertising, website and product sources. You can finally see that the same person who saw three banners registered and stayed into a second month.

Data source	Recorded event	Metric for analysis
DSP	3 advertising contacts	Impression log (ID:123)
Product	Registration, 30-day retention	Retention cohort (ID:123)

These were never two different customers. It was one user, and you were seeing half the picture.

Yes, it is expensive. Integration means pipelines, ETL and API work. Now price the alternative: millions spent on acquisition without knowing who stays. Where DSP budgets run into hundreds of thousands, an error of scale is not a mistake, it is a write-off.

The hardest part is not the ETL. It is the cultural shift — getting marketing and product to treat the same person as one story rather than two disconnected metrics.

Conclusion

The question is how long you are willing to tolerate the asymmetry, when someone else's advertising platform knows more about your customer than your own product.

Some will say it is fine because impressions are growing. That is the trap: growth on the chart is not growth in the business. It is a company whose servers are running while customers are not logging in.

Three doubts:

4. "What if the integration does not work?" Then at least you will learn where you are losing customers.
5. "But the DSP already shows a lot." Enough for advertising. Not enough for the product.
6. "Everyone does it this way." Which is exactly why those who do not have an advantage.

The answer is not another analytics dashboard. It is a bridge between acquisition and retention — linking DSP data to product analytics, even through manual cohort tables, API integrations or user interviews.

Then compare the product map with what the demand-side platforms already know. If the two match, you have a foundation. If they do not, you have a starting point.