

	EPISODE 37 Analysis How to Stop Fraud in 30 ms? Faster Than a Bee Flaps Its Wings. SEPTEMBER 28, 2026 <i>This transcript was auto-generated and may contain errors or inaccuracies.</i>
---	---

Jeff Welcome to "What the F Happened? Fraud and Financial Crime Deconstructed," a DEFEND podcast where we break down what's actually happening across fraud scams, AML and financial crime.

Emily Each episode cuts through the noise to explain the tactics, trends, and real world impact behind the headlines so you're better prepared for what comes next. Let's get into it.

Emily Picture this you are standing in a crowded coffee shop. You hold your phone near the payment terminal. You hear that little beep and a green check mark instantly appears.

Jeff Right. It's just instant.

Emily Yeah. You grab your latte and walk out. The whole interaction took maybe half a second, but in the first thirty milliseconds of that half second, literally faster than a bee flaps its wings, an invisible, high stakes infrastructure war just waged across the globe to prove you are exactly who you say you are. Which is wild to think about.

Jeff It is deciding if that coffee purchase is actually you, or, say, a fraudster halfway across the world who just cloned your digital wallet. It's not about having one really smart algorithm. It's an incredibly brutal engineering constraint.

Emily Which is exactly our mission. For this deep dive, we are uncovering the hidden architecture that makes that instantaneous green checkmark possible for you.

Jeff Yeah, we're getting to the plumbing.

Emily Exactly. We're pulling from two incredible playbooks on building high speed, high throughput fraud prevention for enterprise payment systems. And to put enterprise into context here, we aren't just talking about processing your one latte.

Jeff No, not at all.

Emily We are talking about systems processing upwards of fifteen thousand queries every single second without dropping the ball on security.

Jeff The scale is really what makes it a fascinating problem. I mean, you can build the most sophisticated, accurate machine learning model in the world, but if it takes two seconds to run, it is completely useless in a live payment authorization path.

Emily Right. Because nobody wants to wait.

Jeff Exactly. The merchants gateway would time out, or you'd just be standing at that coffee counter staring at a spinning wheel on your phone. High speed fraud prevention is rarely about the model itself. It's this massive infrastructure challenge where every single microsecond is just heavily taxed.

Emily Okay, let's unpack this because the math here is wild.

Jeff It doesn't seem Impossible.

Emily It doesn't when you look at the latency mandate. This finite budget of time the system is allowed to consume a standard benchmark is thirty milliseconds.

Jeff Sometimes up to one hundred milliseconds for a massive enterprise. But yeah, let's use the thirty millisecond window, right?

Emily So every single step in the fraud decision has to fit inside that tiny sliver of time.

Jeff It's an absolutely unforgiving budget. Every component is fighting for a fraction of a millisecond, right?

Emily But look at the typical time allocations required for a risk decision. You've got event ingestion and normalization, basically receiving the messy data payload and cleaning it up. That's taking five to ten milliseconds.

Jeff Yeah. Just to get the data ready.

Emily And then data retrieval and enrichment, fetching the historical context, that's another fifteen to twenty milliseconds.

Jeff We're already almost at the budget.

Emily Exactly. Then feature computation, fifteen to twenty milliseconds. Then the actual detection and scoring by the machine learning models takes twenty to thirty milliseconds.

Jeff Which is a heavy lift.

Emily It is. And then add in decisioning logic for five to ten and response handling for another five to ten. If you just do basic addition on those maximums, you are at over one hundred milliseconds.

Jeff Yeah, the math just fails.

Emily So how on earth does that physically fit into a thirty millisecond window? It feels like, um, like trying to run a formula one pit stop, but it only works if the mechanic changing the tires, the person fueling the car and the engineer adjusting the aerodynamics are all occupying the exact same physical space at the exact same time.

Jeff That's fascinating. Here is how the architecture solves that mathematical impossibility through structural design. You just hit on what engineers call the thirty millisecond rule.

Emily Okay.

Jeff What's that? It means a risk decision is only as fast as its slowest component. To make your F1 pit stop analogy work, the tire changer cannot wait for the Fueller to finish before they start their job. The system requires absolute ruthless parallel execution.

Emily So everyone working concurrently, no waiting in lines.

Jeff Non-blocking architecture is the technical term for non-blocking.

Emily Got it.

Jeff Yeah. Historically, older systems would receive the transaction data. Then go look up the user's history, then evaluate the static rules, and finally run the machine learning model.

Emily So entirely sequential, right?

Jeff One step finishes, the next begins. But in a sub thirty millisecond world, the moment the system starts parsing the incoming transaction, it is simultaneously firing off requests.

Emily Just all at once.

Jeff Exactly. It's evaluating behavioral analytics, triggering multiple rule sets, and running unsupervised machine learning models all at the same time. The decision engine just acts as a conductor.

Emily Oh, like catching everything at the end?

Jeff Yes, it sits at the end, catching all those independent signals, the exact microsecond they finish combining them and returning a verdict.

Emily So we aren't speeding up one assembly line. We are building five independent assembly lines that operate simultaneously.

Jeff That's a great way to picture it.

Emily But here's the physical limitation I keep coming back to. If all these systems are running at the exact same time, where is the data actually coming from?

Jeff Yeah, that's the big hurdle.

Emily Because you can't just pause and wait for a traditional database to spin up and load your entire five year purchase history, right?

Jeff You absolutely can't. And that is the exact point where legacy systems fall over. They try to query a traditional relational database that was fundamentally built for batch analytics.

Emily It's like end of day reporting kind of stuff.

Jeff Exactly. Hitting these latency targets requires an entirely different approach. Yeah. You need a real time data layer.

Emily Meaning what exactly?

Jeff This means relying on an in-memory architecture. The most critical, frequently accessed data isn't sitting on a hard drive somewhere. It's loaded right into the system's active memory, ready to be read instantly.

Emily Which enables what the playbooks call in-flight feature computation. Right. The system calculates dynamic signals right as the event is happening, rather than relying on stale data.

Jeff Think about transaction velocity. If you use your credit card three times over the course of four days, that is totally normal behavior. But if that exact same card attempts transactions at dozens of different merchants within three minutes, that is a massive red flag. The fraud signal isn't the static credit card number. The signal is the sequence itself.

Emily Wow.

Jeff And if the system waits for an overnight batch process to update your profile and say, hey, this user is suddenly very active, the fraudster has already drained the account.

Emily Because they're long gone.

Jeff Right? The system has to compute that velocity feature in flight literally as the swipe is occurring.

Emily But let me push back on this architecture for a second.

Jeff Sure.

Emily Go ahead. If you are keeping all this dynamic high velocity data like rolling transaction windows, device changes, merchant interactions, if you're keeping all of that in memory for millions of transactions happening simultaneously around the world.

Jeff So a lot of data.

Emily Doesn't a centralized memory bank just become a massive traffic jam? Instead of scaling vertically with one giant expensive server, wouldn't you have to shard that data horizontally across edge nodes based on user ID to prevent the system from locking up?

Jeff That is a great point. And yes, sharding the data horizontally is the only way to survive. Centralized state is basically a death sentence at enterprise scale.

Emily Really a death sentence?

Jeff Oh yeah. If you have fifteen thousand transactions a second, all trying to read and write to one single in-memory cluster, the database locks up latency spikes and the entire system crashes.

Emily So how do you distribute the state effectively?

Jeff Okay, so instead of one massive library desk where fifteen zero zero zero people are shouting at one librarian to check their records.

Emily Which sounds like a nightmare.

Jeff It is. So instead of that, you divide the records alphabetically into a thousand different rooms, each with its own librarian by partitioning the data across numerous separate nodes, maybe hashing it by account ID. The memory holding the state for one group of users is physically separated from another.

Emily Okay, that makes sense.

Jeff The load is spread perfectly across a vast horizontal plane when transaction volume increases. You don't buy a bigger centralized machine, you just add more nodes to distribute the state even further.

Emily Which prevents any single component from becoming the bottleneck. But speaking of fifteen zero zero zero people shouting, what happens to this finely tuned millisecond perfect ecosystem when a massive surge hits?

Jeff Uh. The spikes.

Emily Yeah. Say it's Black Friday or a major new product drops, and suddenly traffic just spikes aggressively handling fifteen zero zero zero plus queries per second is the baseline, but bursts can far exceed that, right?

Jeff They absolutely can. And bursts introduced the degraded decisioning trap, which is an incredibly dangerous state for a payment processor.

Emily Degraded decisioning.

Jeff Yeah. When a poorly designed system gets overwhelmed by a massive spike in volume, its latency goes through the roof to avoid crashing the entire authorization network, the system essentially panics.

Emily Oh no.

Jeff It enters a survival mode where it starts skipping the hard, computationally expensive work. It will bypass heavy machine learning models, skip complex knowledge graph lookups, and just fall back on simple static rules.

Emily It's like a bouncer at a really busy nightclub, you know?

Jeff Ha. Yeah, exactly.

Emily Early in the night, he's meticulously checking every ID, tilting it to look at the hologram, matching the face to the photo.

Jeff Taking his time.

Emily But then a massive party bus pulls up. Two hundred people rushed the door all at once, and the bouncer just panics and starts waving everyone through because the line is backing up into the street.

Jeff And the danger of that party bus scenario is that fraudsters actually manufacture those crowds on purpose.

Emily Wait. Really? On purpose?

Jeff Oh, absolutely. They know exactly how to operate the bus. They use techniques like card testing, mass credential abuse or coordinated bot attacks to intentionally generate massive transaction volumes.

Emily Just to flood the system.

Jeff Yes, the sheer volume isn't just an accident. The goal is to purposefully overwhelm the processor's infrastructure, forcing the system into that degraded, dumbed down state.

Emily Wow.

Jeff Once the bouncer starts waving people through, the high value, fraudulent transactions slip right past the resulting detection gap.

Emily So they are weaponizing the system's own capacity limits against itself. That is brilliant and terrifying.

Jeff If we connect this to the bigger picture, this is exactly why the industry's reliance on average latency metrics is a complete illusion. You will hear infrastructure vendors boast, oh, our average latency is fifty milliseconds.

Emily Which sounds good, right?

Jeff It sounds great, but the P50, the median average is a really dangerous metric to rely on. A fifty millisecond average is utterly useless if your p ninety five or P ninety nine tail latency is five hundred milliseconds.

Emily Because that tail latency represents the moments when the system is under stress. Exactly during peak spikes. A five hundred millisecond delay means the system is grinding to a halt.

Jeff Yes, and in payments, five hundred milliseconds means transactions are actively timing out at the gateway level. The authorization fails, legitimate customers get declined, and the business loses revenue.

Emily Right.

Jeff A true enterprise architecture cannot be engineered for a calm average Tuesday afternoon. Capacity planning has to be engineered for peak traffic in sudden bursts.

Emily It has to survive the party bus.

Jeff It has to process that fifteen thousand queries per second, with the full intelligence stack running without ever degrading the decision.

Emily So assuming an organization has secured the speed and they've built the horizontal scale to handle burst traffic without dumbing down the bouncer, how smart is the system, actually?

Jeff That's the real test.

Emily Right. Because the intelligence seems to hinge entirely on connecting a massive spider web of data across the financial ecosystem in real time. This concept of dynamic entity Resolution.

Jeff Entity resolution is what separates a basic rules engine from true fraud prevention. I mean, a single transaction evaluated in a vacuum tells you almost nothing.

Emily It's just a.

Jeff Coffee. Exactly. But what if the system instantly recognizes that the IP address used for your coffee purchase was also seen five minutes ago? Systematically trying to access ten different bank accounts.

Emily Yikes. Okay.

Jeff Or what if the device ID matches a phone known for initiating fraudulent wire transfers last month?

Emily This is where bypassing those knowledge graph lookups during a traffic spike becomes so fatal. Because the knowledge graph is essentially the map of how your device, your IP, and your account are historically linked to other users. Right.

Jeff Exactly. Entity resolution traverses that graph instantly. It connects devices, accounts, IPS, counterparties, and merchants. It reveals coordinated network level fraud that is completely invisible. If you only look at one isolated card swipe.

Emily And the key is applying this intelligence across what they call one engine, every rail. Meaning this risk context overlaps across credit cards. ACH transfers, wires, real time payments, fednow, peer to peer apps, digital wallets, literally everything.

Jeff Everything.

Emily Because a device used in a shady card swipe today might show up in an ACH account takeover tomorrow.

Jeff Fraudsters do not silo themselves to one single payment method, so the risk infrastructure cannot operate in silos either. Having a shared risk view across every rail is critical.

Emily Here's where it gets really interesting, though. We are mixing vastly different payment rails here. ACH takes hours or days to settle in batches. Wires are traditionally slow but handle massive high value transfers. Real time payments and fednow are instant bank to bank settlements, right?

Jeff Very different profiles.

Emily So if I do something slightly unusual on a peer to peer app, maybe I send my friend money for dinner, but I'm traveling and on a weird public Wi-Fi network. Does that shared risk view automatically block an entirely separate, massive wire transfer I need to make for a house down payment later that day?

Jeff That would be awful, right?

Emily How does this interconnected web avoid becoming a nightmare of false positives where I just can't touch my own money?

Jeff That is the ultimate tension in fraud prevention. It's balancing security with customer friction. But there is a very critical distinction to make here the difference between detection and decisioning.

Emily Okay.

Jeff Unpack that. Detection is the act of identifying the risk context. Decisioning is the business logic of what you actually do about it. While the risk context is shared broadly across all rails, the execution the decision is strictly rail specific.

Emily Ah I see. So the system knows it's all me, and it shares the context that my device is on a strange network, but it treats the action differently based on the specific type of payment I'm attempting.

Jeff Precisely. Remember our latency budget in the rules of the rails, a card swipe at a coffee shop as a sub one hundred millisecond service level agreement. It has to happen instantly, right? If the system detects slight risk, it might just decline the card to be safe because the friction of a decline is low. You just try another card.

Emily Or an annoyance.

Jeff But a high value wire transfer for a house down payment that operates on a totally different payment rail. With a much longer SLA, the system has more time to decide.

Emily Okay.

Jeff So it uses the shared risk context from your peer to peer app, notes the unusual Wi-Fi. But instead of automatically blocking the wire, it triggers a step up authentication.

Emily Oh, so it sends a push notification to your phone asking you to verify with biometrics.

Jeff Yes, precision comes from having the shared context, but applying the tailored action for the specific payment rail.

Emily That makes perfect sense. It's tailored friction, but orchestrating all of this distributed in-memory state parallel execution Cross rail entity resolution via knowledge graphs. It sounds incredibly daunting.

Jeff It's no small feat.

Emily How does an organization actually know if their current infrastructure is fundamentally broken? Because looking at the common infrastructure blueprint, there are really eight common bottlenecks that kill real time performance.

Jeff Bottlenecks are the silent killers of scale. Some of the most common culprits are remote data dependencies, where the system pauses to fetch data from a server located in an entirely different geographic region.

Emily Which eats up the budget.

Jeff Exactly. And centralized state, which we covered with the traffic jam scenario. You also see sequential detection where models politely wait their turn instead of running in parallel.

Emily Right? The blocking architecture.

Jeff Yep. An expensive feature computation that forces the system to crunch massive historical data sets rather than using in-flight signals.

Emily Then there is uneven load distribution, infrastructure contention, and synchronous downstream dependencies.

Jeff That last one is a big one.

Emily It's huge. I mean, if your system stops and waits for a third party credit check to return a score before moving to the next step, and that third party has a minor outage, your entire thirty millisecond budget instantly evaporates.

Jeff It's gone.

Emily It brings me to this realization. A latency problem is almost always a system problem. It is completely useless for a data science team to spend six months optimizing a machine learning model to be ten milliseconds faster. If the API routing calling that model is broken and adds fifty milliseconds of network delay.

Jeff This raises an important question how do you diagnose that routing delay in a live environment? You cannot fix what you cannot see. The biggest mistake organizations make is limited observability. Oh, so they only measure latency at the final API boundary, the very end of the line where the final decision is handed back if that final number is high. Leadership just assumes the machine learning model is too slow.

Emily But the model might be fine. The delay could be the remote data retrieval or the Entity resolution graph or a synchronous queue.

Jeff Backing up true observability means instrumenting and measuring the latency of every single micro component in the critical path. And that is what the first thirty days of a proper ninety day infrastructure readiness plan is all about. Mapping the complete transaction path.

Emily Just seeing where everything goes, you.

Jeff Have to trace a request through every microservice to measure end to end latency and throughput at every stage. You have to know exactly which component is bleeding out your millisecond budget.

Emily And once you map it. Days thirty one to sixty are for identifying the specific constraints like those data access or state management issues. Right? But the key is doing this using actual production traffic, not just

synthetic lab tests, because as we discussed, the party bus only shows up in production. You can't simulate the true chaos of a bot attack in a sterile lab environment.

Jeff No, it never behaves the same way.

Emily You have to route shadow traffic through the new architecture to see where it chokes.

Jeff Shadowing production traffic is essential. And finally, days sixty one to ninety are dedicated to stress testing against peak volume and deliberately breaking dependencies.

Emily Chaos engineering.

Jeff Exactly. Chaos engineering. You intentionally degrade a downstream service, like simulating a timeout from that third party credit check, just to prove that your parallel architecture can handle the failure gracefully without crashing the entire authorization flow.

Emily It's a rigorous, almost combative framework, but for processing fifteen thousand queries per second, it's the only way to ensure the architecture holds up when the pressure is on.

Jeff You have to test it until it breaks.

Emily So what does this all mean for you listening right now? We've journeyed deep into the anatomy of a payment. Blink. True, high speed fraud prevention isn't about hoping one magic algorithm works fast enough.

Jeff Not at all.

Emily It is an architectural marvel. It requires orchestrating strictly budgeted, in-memory, distributed data. It demands parallel non-blocking execution where five complex evaluations happen simultaneously, right? And it relies on Crossrail entity resolution to connect the dots instantly delivering thousands of decisions a second without dropping the ball on intelligence or falling back to dumb down rules when the traffic spikes.

Jeff It is a masterclass in distributed systems engineering. Yeah, the underlying architecture have to be as resilient as it is incredibly fast.

Emily Which leaves me with this final thought for you.

Emily To mull over. If enterprise fraud models now rely on continuously updating in-flight behavioural state, a highly intelligent system that spans every payment method we use, instantly resolving our devices, our locations, and our habits in literally milliseconds. Yeah. Will we eventually reach a point where traditional credit scores or static identity checks become completely obsolete, replaced instead by a millisecond by millisecond behavioral pulse that seamlessly determines our financial trustworthiness in real time. Next time you tap your phone for that coffee and see the green check mark, just think about the invisible pulse you just sent into the machine.

Jeff You've been listening to "What the F Happened? Fraud and Financial Crime Deconstructed," a DEFEND podcast by DataVisor. If you want to keep learning between episodes, check out DEFEND Training, a set of self-paced online courses for fraud and financial crime professionals, practical and built around real world scenarios.

Emily And you can earn CPE credits through the ACFE San Francisco Bay area chapter. You can find it at datavisor.com/defend-training. The link is in the description.