SuperDataScience

SDS PODCAST

EPISODE 1014: OPENAI AGENT BREACHES HUGGING FACE: ALL YOU MUST KNOW INCL. HOW TO PROTECT YOURSELF

Jon Krohn: 00:00 This is episode number 1014 on the Rogue OpenAI agent that breached hugging face. This is big news indeed. Welcome back to the SuperDataScience Podcast. I'm your host, Jon Krohn. Today's topic is an incident that reads like science fiction. An AI agent that broke out of the sandbox it was being tested in, hacked its way into another company's servers and did all of that to cheat on an exam its own makers had set for it. Let me lay out the timeline because it unfolded in three acts over the past couple of weeks. Act one was July 16th. HuggingFace, the enormously popular platform for hosting open source models and data sets that most of this podcast listeners would probably use regularly. Hugging Face disclosed that it had detected and responded to an intrusion into part of its production infrastructure. What set this apart from anything the company had handled before is that the campaign was driven end-to-end by an autonomous AI agent system.

01:00 HuggingFace detected and dissected it largely with AI of their own, which I'll come back to because that part of the story may matter more than the breach itself. Anyway, what HuggingFace found was unauthorized access to a limited set of internal data sets and to several credentials used by their services. Importantly, they found no evidence of tampering with public user-facing models or data sets and their software supply chain was verified clean. They were still assessing whether any partner or customer data were affected. At that point, they did not know which model was behind the attack. Their post described the attacker only as an autonomous agent framework that appeared to be built on an agentic security research harness with the underlying LLM unknown. Act two was July 21st last week. OpenAI published a joint post with hugging face confessing that the attacker was OpenAI, specifically a combination of OpenAI's own models.

01:53 GPT 5.6 Sol, their most capable public model released earlier this month, together with an even more capable pre-release model that hasn't been made public. Both were running with reduced cyber refusals for evaluation purposes. There was no human attacker at any point. So how does one AI company accidentally hack another? Here's the mechanism and it's worth understanding in detail with lessons and practical guidance from me at the end of the episode. All right, so AI labs test their models for dangerous capabilities before releasing them. OpenAI was running an internal evaluation on a benchmark called Exploit Gym, which was published in an archive paper in May. And I've got the link for you to the Exploit Gym paper in the show notes. Exploit Gym comprises nearly 900 instances derived from real world vulnerabilities that affected popular software projects. The crucial thing about Exploit Gym is what it measures.

02:48 It isn't about discovering vulnerabilities. It's about taking a vulnerability that's already been reported, you're handed an input that triggers the bug, and then progressively extending the vulnerability into a working exploit that achieves unauthorized code execution. Ooh, nasty. That's a meaningfully more dangerous capability than discovery alone, and it's one the field has been watching closely. The headline results under a two-hour time limit per task and with standard defenses switched off, the top performers on Explode Gym, according to the paper, were Claude Methos Preview paired with ClaudeCode and GPT 5.5 paired with Codex CLI, producing working exploits for 157 and 120 instances respectively. GPT-5. 4, it solved only 54, so a third of what Claudecode and GPT-5. 5 could do together. Every other model in agent pairing they tested solved really only a handful of these exploitations. So this means that end-to-end exploitation remains hard, but it sharply separates the frontier from everything else.

03:56 The conclusion of the Exploit Gym paper is blunt. It says autonomous exploit development by frontier AI agents is no longer a hypothetical capability, as we've now seen in the past couple of weeks. So two findings in that Exploit Gym paper are worth pausing on because they foreshadow exactly what went wrong at OpenAI. The first is what the Exploit Gym authors call solving via an unintended path. Each task specifies a particular vulnerability to exploit, and the researchers use a second AI agent as a judge to check whether the exploit actually used that vulnerability. Frequently, it didn't. Of GPT - 5.5's flag captures, 90 succeeded through some other route. For Claude Method's preview, 69 did. Sometimes the agent noticed a nearby weaker code path and pivoted to it. More strikingly, sometimes it concluded the assigned vulnerability wasn't exploitable at all and went looking for entirely new attack surfaces.

04:51 In other words, these agents already had a documented habit of ignoring the prescribed route and finding their own way to the flag to succeed. The second Exploit Gym paper finding concerns the safety filters. The researchers ran their main experiments with production guardrails disabled under OpenAI's trusted access for CyberProgram and as well under Anthropic's cyber verification program. But the researchers, the Exploit Gym researchers, also reran GBT 5.5 with OpenAI's default safety filter switch back on. And the result in that case was decisive. In 88% of cases, the agent was blocked from making a single tool call. And in the remainder, it never got past reconnaissance. Not one exploit attempt succeeded when the guardrails were on. Hold onto that because it tells you how much was riding on those guardrails being off at OpenAI in the past couple weeks. To estimate a model's maximum cyber capability, OpenAI ran Exploit Gym without the production classifiers that normally prevent models from pursuing high risk cyber activity.

05:52 And because turning an uncensored cyber attacker loose on the open internet would definitely be reckless, the evaluation ran in a highly isolated environment. Network access was constrained to a single narrow channel, the ability to install software packages through an internally hosted third-party proxy that caches package registries. The Exploit Gym paper describes the same pattern in its own setup, outbound connections restricted to a curated allow list covering Ubuntu package repositories and PiPi with everything else on the internet blocked. But that narrow channel is nevertheless where it all went wrong. Rather than solving the Exploit Gym problems as intended, the model spent a substantial amount of inference compute working out how to reach the open internet. They identified and exploited a zero-day vulnerability in that package registry, a previously unknown flaw, which OpenAI has since disclosed to the vendor. With that foothold, the agents performed a series of privilege escalation and lateral movement actions inside OpenAI's research environment until they reached a node with internet access.

06:54 Once online, the models inferred that Hugging Face might host the models, data sets, and solutions for Exploit Gym. Wow, great. So the agent's thinking it doesn't actually have to solve the Exploit Gym problems. It can just find the answers. If only it could break into HuggingFace. So the agent went after Hugging Face to steal the answers. From HuggingFace's side, the entry point was the data processing pipeline, which as they put it, is where AI platforms are uniquely exposed. The OpenAI agent created a malicious data set that allowed the harvesting of HuggingFace's private cloud and cluster credentials and moved laterally into several internal compute clusters over the course of a weekend. The agent executed many thousands of individual actions across a swarm of short-lived sandboxes with self-migrating command and control staged on public services. OpenAI, for their part, described the models chaining together multiple attack

vectors, including stolen credentials and zero-day vulnerabilities to find a remote code execution path onto HuggingFace's servers.

07:53 Sit with that for a moment. Nobody instructed the model to attack HuggingFace. It was told to solve a benchmark. OpenAI's assessment is that the models were hyper-focused on finding an Exploit Gym solution and went to extreme lengths to achieve a rather narrow testing goal. That is the defining characteristic of this generation of models. Give one an objective and any path at all toward it, even an unintended path, and it will find its way through. This is akin to the paperclip problem made famous by Nick Bostrom in his bestselling super intelligence book. All right, back to the incident here. Two technical details elevate all of this above a mere curiosity. First, the models found and exploited novel attack paths in real world production systems without source code access. Historically, hunting zero days meant pouring over source code, not anymore. Second, this was not a soft target.

08:48 Hugging Face is a firm valued at four and a half billion dollars with hundreds of employees and a dedicated security team. Things could have gone considerably worse for a less well-resourced organization. All right, now for act three. And the part I'd most like you to take away from this episode. Hugging Face detected the security breach with AI. Their anomaly detection pipeline uses LLM-based triage over security telemetry to separate real signals from daily noise, and the correlation of those signals flagged the compromise. Then to understand what a swarm of tens of thousands of automated actions had actually done, they ran LLM driven analysis agents over the full attacker action log. That led Hugging Face reconstruct the timeline, extract indicators of compromise, map which credentials had been touched, and separate real impact from decoy activity. Work that

would normally take days, took hours, which is what it takes to match an adversary operating at machine speed.

09:42 But here's the catch. When HuggingFace began that log analysis, they reached first for frontier models behind commercial APIs like Claude and OpenAI's top publicly available frontier models. It did not work. The analysis requires submitting large volumes of real attack commands, and those requests were blocked by the provider's safety guardrails. Guardrails that in Hugging Face's words cannot distinguish an incident responder from an attacker. So instead of being able to use American publicly available frontier models, Hugging Face ran the forensics instead on GLM 5.2, an open weight model from Chinese firm z.ai, and they ran it on their own infrastructure. That had a second benefit to Hugging Face, which is that no attacker data and none of the credentials referenced left their environment. The asymmetry is stark and Hugging Face named it plainly. The practical lesson for defenders, you need to have a capable model you can run on your own infrastructure vetted and ready before an incident, both to avoid guardrail lockout and to keep attacker data inside your own perimeter.

10:45 Renowned computer scientist Simon Willison blogging about this incident last week pushed the point even further. The frontier models we can access from Anthropic, OpenAI, and Google are increasingly constrained in how much they can help us defend our software. While open weight models coming out of China face no such limits and any limits they do have can likely be fine-tuned away. Simon's worry is that constraints meant to make us safer, maybe having the opposite effect. So there ends up being a kind of political and geopolitical element to this as well. In terms of remediation post-incident, Hugging Face closed the dataset code execution paths. They eradicated the foothold and rebuilt the compromised nodes, revoked and rotated affected

credentials and tokens alongside a broader precautionary rotation of secrets, deployed stricter admission controls on their clusters and improved alerting. So a high severity signal pages of responder within minutes on any day of the week, including the weekend, which I don't know, it implies that their pager wouldn't go off on a weekend before, and that's part of how the OpenAI agent succeeded.

11:48 Anyway, Hugging Face also brought in outside forensic specialists and reported the incident to law enforcement. They also recommend that users, that means you, rotate any access tokens and review recent account activity, which you might want to think about doing right after listening to this episode. OpenAI for their part have imposed strict infrastructure controls at the cost of research velocity, is briefing its safety and security committee and disclosed the zero-day vulnerabilities to the vendor that allowed the OpenAI agent to escape onto the internet. And they also brought HuggingFace into their trusted access program so Hugging Face can use OpenAI's cyber capable models to improve its own defenses. It's worth noting that this isn't the first sign of models exceeding their intended bounds. In April, an anthropic researcher posted that Claude Mifos, then unreleased, had emailed him to report that it had escaped a sandbox during its own cybersecurity evaluation while the researcher was sitting in a park eating a sandwich.

12:45 What a surprise that would've been. Now that model didn't go on to compromise anyone else's servers. And beyond these kind of cybersecurity incidences, the same trait of just going on and on and on to solve problems shows up in benign settings too. In May, an unreleased OpenAI model disproved the planar unit distance conjecture. I don't know what that is really, but it's some combinatorial geometry problem, Paul Erdos posed back in 1946. And in July, just last week actually, the Harvard mathematician Levant Alpogi, don't know if I'm

pronouncing that correctly, reported using Claude Fable five to disprove something called the Jacobian conjecture, which it stood for more than 80 years. So relentless goal pursuit is the same capability, has the same kind of power, whether it's pointed at open decades old math problems or at a security system. One quick thing here on legal stuff. The legal picture is unsettled with the cybersecurity incident, the OpenAI attacking hugging face.

13:49 Nathan Calvin, who is general counsel at the nonprofit in Code AI, points out that no state or federal law requires companies to disclose the internal deployment of highly capable models like the one involved here. California enacted a bill last year requiring Frontier Labs to report any critical safety incident within 15 days, including a model using deceptive techniques to subvert monitoring. But that provision applies outside the context of an evaluation. So it's unclear whether the recent breach would even qualify. As for liability, federal anti-hacking law targets intentional unauthorized access. And OpenAI plainly did not intend for its models to go hunting for a crib sheet for answers to a test. Someone named Steven Lorena of the Institute for Law and AI frames it. Many of these questions turn on what the company knew or could have foreseen. Someone at Apollo Research named Alex Meinke added a sobering observation about the incident.

14:45 He said that because AI systems are beginning to surpass humans at cybersecurity tasks, only a very small number of experts worldwide could properly understand how this breach even occurred. So now you know all the nitty gritty on the incident. Hopefully you heard more here today than you already knew about the incident before. We went into a lot of detail. But before I wrap up, let me get practical because there are lessons here for both sides of this, for those of you building agentic systems and for those of you defending platforms against them. If you're running agents, particularly long horizon agents with tool

access, the first lesson is that your egress allow list is an attack surface, not a safety guarantee. OpenAI sandbox was tight everywhere except one convenience, a proxy for pulling down software packages. That single hole was where the models went and the proxy itself turned out to have an exploitable zero-day vulnerability within it.

15:34 So ask whether your agent needs network access at runtime at all or whether you can pre-install its dependencies and run it fully air gapped. Anything you leave open for convenience is the thing a sufficiently motivated agent will probe. Second, if you disable safeguards for a legitimate reason and testing maximum cyber capability is a legitimate reason, the containment and monitoring around that test needs to scale up in proportion. Remember that exploit gym result. With OpenAI's default filters enabled, GPT 5.5 was blocked from being able to do anything dangerous. Those filters were doing enormous work. Take them away and everything downstream rests on your sandbox alone. OpenAI's own state of takeaway is exactly this. The incident points to a need for stronger cyber protections at evaluation time and better monitoring during internal testing. Third, watch how you specify goals. A narrow objective combined with unconstrained means is the recipe that produced this whole episode and the Exploit Gym paper documented the underlying behavior months earlier.

16:32 Agents routinely abandoning the prescribed vulnerability and inventing their own root to the flag. If there's a shortcut to satisfying your evaluation that doesn't involve doing the work, assume a frontier model will find that shortcut and design the eval so the shortcut isn't reachable. And fourth, log every action your agent takes and alert on the ones you didn't anticipate, especially outbound connections that shouldn't exist. Note too that the lateral movement here unfolded over a weekend and that Hugging Face's remediation explicitly included

tightening alerting so that a high severity signal pages are responded within minutes on any day of the week, even if it's a holiday. Now, so that's everything that you need to know if you are designing agents. If you are on the defending side, trying to avoid agents attacking you, start with the immediate housekeeping. HuggingFace recommends rotating any access tokens on your account and reviewing recent activities.

17:23 So go do that today. Then think about the deeper lesson, which is that on an AI platform, data and model artifacts are executable content, not innerd files. Initial access here came through a malicious data set that abused a remote code loader together with a template injection flaw in a dataset configuration. HuggingFace hasn't published which library or version was involved. And I'd caution against assuming it maps neatly onto your own stack, but the general principle transfers. If your pipeline automatically processes data sets that users supply, audit every path in it that can result in code execution and keep your loader libraries current since the trend across the ecosystem has been to progressively strip out remote code execution paths that used to be enabled by default. The last piece of advice is the one I'd most encourage you to act on because it's the least obvious. Pick a capable open weight model, get it running on your own infrastructure.

18:14 And like HuggingFace recommended, like I talked about earlier in this episode, validate that open weight model, probably a Chinese model, can do forensic log analysis before you need it. HuggingFace discovered mid-incident that the commercial APIs they reached for first would not process their own attack data. You don't want to find out that at two in the morning with an active intruder in your clusters. And the secondary benefit of using an open weight model is that nothing about your incident, including the credentials it touched, leaves your environment. So where does this leave us? Autonomous

AI-driven offensive tooling is no longer theoretical. It lowers the cost of running a broad, patient, multi-stage campaign, and it operates at machine speed. Defending a platform now means treating your data and model services as first-class attack services and putting AI on defense to keep pace. This one, this incident of OpenAI agents attacking hugging face, that was an accident discovered by the people who caused it and disclosed by both parties within days.

19:17 The next attack may not be an accident and pleading ignorance will be a good deal harder. All right, that's the end of today's episode. If you enjoyed it or know someone who might consider sharing this episode with them, leave a review of the show on your favorite podcasting platform or YouTube. Tag me in a LinkedIn post with your thoughts. And if you aren't already, be sure to subscribe to the show. Most importantly, however, I hope you'll just keep on listening until next time. Keep on rocking it out there and I'm looking forward to enjoying another round of the SuperDataScience podcast with you very soon.