Weather in Lawrenceburg: 54.4°C, Mainly clear

Daily Rosary

****

The Joyful Mysteries

Annunciation, Visitation, Nativity, Presentation at the Temple, Finding in the Temple

Local News - Top Stories

****

1. Over 250 Vehicles Enter Cruisin to a Cure for ALS Car Show Despite Rain

Donnie Hastings Jr. took Best of Show this year.

2. UPDATE: Short-Term Closure Planned on State Road 129 in Ripley County

The scheduled start date has been pushed back.

3. Southeastern Career Center Students Commit to Future Plans

Twenty-seven students committed to furthering their education or entering the workforce.

4. All-Way Stop, Speed Limit Change Coming to State Road 46 in New Point

5. Volunteers Needed for Litter Clean-Up on York Ridge Hill

The clean-up is scheduled for May 10.

Hacker News - Top Stories

****

1. AWS Built a Security Tool. It Introduced a Security Risk

AWS Built a Security Tool. It Introduced a Security Risk. BlogApr 28, 2025 | 7 minAWS Built a Security Tool. It Introduced a Security Risk.Eliav LivnehSecurity Researcher(If you missed the previous parts of this trust policy blog series, we recommend reading parts one and two first)In the previous post of this series, we explored four dangerous misconceptions regarding how to securely set up cross-account access in AWS environments.In this final post of the series, weâ■■ll walk through a real-world case where even AWS got it wrong. Their Account Assessment for AWS Organizations tool, designed to audit resource-based policies for risky cross-account access, ironically introduced cross-account privilege escalation risks due to flawed deployment instructions. Specifically, customers were effectively encouraged to deploy the tool in lower-sensitivity accounts, creating risky trust paths from insecure environments into highly sensitive ones.Weâ■■ll share how we discovered the issue, the risks it introduced, how AWS fixed it, and what affected organizations should do to detect and remediate it.How it startedWhile investigating a critical privilege escalation risk involving an IAM role in a customerâ■■s AWS environment, we discovered a role present in both their production and management accounts, each of which trusted two roles in their development account:The risky IAM role we investigatedThese were the details of the privilege escalation risk (sensitive info redacted):Examining the permissions of that role in the production and management accounts, we found it had access to several sensitive IAM and data-related API calls, including:iam:ListRoles (lists all IAM roles, helping an attacker identify privileged and vulnerable roles)iam:ListPolicies (reveals all IAM policies, exposing potential misconfigurations)secretsmanager:ListSecrets (lists all stored secret names, identifying potential targets)s3:ListAllMyBuckets (enumerates all S3 buckets, exposing potential sensitive data locations)kms:ListKeys (lists all encryption keys, indicating what is being encrypted)kms:GetKeyPolicy (retrieves key policies, which could reveal weak or misconfigured access controls)â■¦ and more (full policy here)These permissions were granted on all resources (resource: "*"), making them particularly dangerous if compromised.The organization's dev account had weaker security, making this role assumption path a major risk. An attacker compromising a trusted role in dev could immediately gain these permissions in production and management. Combined with other misconfigurations - such as exposed IAM roles, secret names, KMS keys, or public S3 buckets - this could help lead to compromise of the organization's most sensitive accounts.This was already a serious risk, but what we found next revealed that this misconfiguration was not unique to this organization - it was actually inadvertently encouraged by AWS itself.Account Assessment for AWS OrganizationsThe names of these roles were somewhat quirky, and hinted they were part of some automated system. A quick search led us to Account Assessment for AWS Organizations, a tool developed by AWS and published in the AWS Solutions Library.According to its official documentation, the tool is designed to â■■centrally evaluate and manage AWS accounts within your AWS Organizationsâ■■, helping users better understand account dependencies. AWS lists its primary use cases as mergers and acquisitions, security audits, centralized policy explorer and management account transitions.So, why was it deployed insecurely?Given that this tool was built by AWS to audit resource-based cross-account access, we initially assumed the misconfiguration was a deployment error by the customer. However, after reviewing the documentation, we realized that AWSâ■■s deployment instructions unintentionally encouraged insecure setups, making it highly likely that users would deploy the tool in a way that introduced privilege escalation risks.The tool follows a hub-and-spoke architecture:A hub role is deployed in a designated hub account.Spoke roles are deployed in all other accounts to be assessed, trusting the hub role for access.This design allows the hub role to assume the spoke roles across all accounts, aggregating security data across the organization.When checking AWS's official deployment guidance, we were surprised to see the following instruction:"Hub stack - Deploy to any member account in your AWS Organization except the Organizations management account."This is the root cause of the issue - AWS explicitly recommended not deploying the hub in the management account, without clarifying the security implications of the other possible choices.Why this is a major security riskLetâ■■s back up a bit and explain why this instruction, without any clarification, was so problematic.As we covered in the previous blog, allowing a role in a less secure account to assume roles in more sensitive accounts creates a privilege escalation risk.By design, the hub role must access all spoke accounts. Since AWS prohibited using the management account as the hub, customers were forced to deploy the hub in a less secure account - often a development, sandbox, or similarly low-sensitivity account. In deployed organizations, this led to the creation of a direct trust path from a lower-security account to higher-security accounts like production, PCI-DSS environments, and even the management account itself.The following diagram visualizes the problem:In the case we investigated, the customer deployed the hub role in their development account, which:(Reasonably) had weaker security controls than production or management.Had full access to assume spoke roles across all accounts, including sensitive ones such as the management and production accountsThis meant that if an attacker compromised the development account, they could pivot into the management account - make it much easier to gain control over the entire AWS organization.AWS strongly advises against running workloads in the management account, so simply deploying the hub there is not an ideal solution either. Instead, the only relatively safe option is to choose an account which is as secure as the management account, such as a centralized DevOps or Infrastructure account with strict access controls.However, some organizations donâ■■t necessarily have such an account, meaning any deployment inherently introduces risk. The problem isn't just where to deploy the hub - it's that AWS's default recommendation pushed organizations into an insecure setup without any reference to the security implications of this choice.Implications for affected organizationsAny organization that deployed this tool following AWSâ■■s instructions (before they were fixed on 2025-01-28) with the hub role deployed in an account less secure than the management account - and hasnâ■■t removed it - is at risk.For affected orgs, the hub account (where the hub role is deployed) becomes a high-value target. If an attacker compromises this role, whether through direct access or privilege escalation, they can assume spoke roles across all linked accounts - including highly sensitive environments such as management and production.Once inside, they can:Enumerate IAM roles, trust policies, and permissions, identifying weak points.List all S3 buckets and secret names, mapping out valuable data targets.Access KMS key policies, which may reveal misconfigured encryption controls.Additionally, the toolâ■■s predefined role names make it even easier for attackers to exploit. If an attacker gains access to any AWS account within an affected organization, they can easily identify whether the tool is deployed and know the exact role names that can be assumed from the hub account. This reduces the effort needed to escalate privileges and pivot into high-value environments.Detecting and remediatingHere are two methods to determine if your organization is affected:Use the AWS Conso

2. The vocal effects of Daft Punk

The vocal effects of Daft Punk Apps Articles Help Contact The vocal effects of Daft Punk Daft Punk have used a wide variety of vocal effects in their songs. A May 2001 interview in Remix magazine provided a rare insight from Daft Punk themselves on the topic. â■■People always ask us what vocoder we use, but every one of our vocal tracks uses a different vocoder effect. We have the old Roland one [an SVC-350], Auto-Tune, and a DigiTech Vocalist.â■■ The quote delivers some vital clues, but itâ■■s incomplete, covering only their first two albums. Thereâ■■s no mention of using a talk box, despite Around The World almost certainly using one. The quote makes it sound like the DigiTech Vocalist is a vocoder, but itâ■■s not. And for that matter, which DigiTech Vocalist model? Thereâ■■s around 30 pieces of hardware in DigiTechâ■■s Vocalist series, and quite a few of them were around before Discoveryâ■■s release in 2001. Iâ■■ve read comments suggesting the DigiTech Vocalist models with the â■■EXâ■■ suffix are special, but nobody seems to know why, and nobody has published a direct comparison to prove or disprove the theory. I decided to take on the challenge, and run the tests myself. Hereâ■■s a fraction of the Vocalist units I ended up buying. Why are there duplicates of the same model? Why is there a Korg ih in that photo? Before this article gets sidetracked with tests and some honestly quite interesting corporate partnerships, mergers, and lawsuits, here is a list of every Daft Punk album song containing robot-like vocal effects, and my guess on which piece of kit was used for the vocals. Album Song Effects Homework WDPK 83.7 FM Roland SVC-350 Homework Around The World Talk box Homework Teachers Ensoniq DP/4+ Homework Oh Yeah Ensoniq DP/4+ Discovery One More Time Auto-Tune Discovery Digital Love DigiTech Vocalist Discovery Harder, Better, Faster, Stronger DigiTech Talker Discovery Something About Us DigiTech Vocalist Human After All Human After All DigiTech Talker Human After All The Prime Time Of Your Life DigiTech Talker Human After All Robot Rock DigiTech Talker Human After All The Brainwasher Tremolo Human After All Television Rules The Nation DigiTech Talker Human After All Technologic Ensoniq DP/4+ Human After All Emotion Roland SVC-350 Random Access Memories Give Life Back To Music Sennheiser VSM201 Random Access Memories The Game Of Love Sennheiser VSM201 Random Access Memories Within Sennheiser VSM201 Random Access Memories Instant Crush Auto-Tune and VSM201 Random Access Memories Lose Yourself To Dance Talker and VSM201 Random Access Memories Touch Sennheiser VSM201 Random Access Memories Get Lucky Sennheiser VSM201 Random Access Memories Beyond Sennheiser VSM201 Random Access Memories Fragments Of Time Talk box (synth solo) Random Access Memories Doinâ■■ It Right Sennheiser VSM201 Homework notes (20 January 1997) # There arenâ■■t many robot vocal effects on Homework, but there is a lot of pitch shifting, likely provided by Daft Punkâ■■s Ensoniq DP/4+, a digital multi-effects units that can do a variety of things. I donâ■■t believe Daft Punk used the vocoder on the Ensoniq DP/4+ for Homework or any of their other albums. The Remix magazine quote says Ensoniq DP/4, but a gear list in another interview says DP/4+. It doesnâ■■t matter which model was used, as the pitch shifting and vocoder sound the same on both units. Discovery notes (12 March 2001) # One More Time sounds like Auto-Tune in combination with a Mu-Tron Phasor or Moogerfooger. Harder, Better, Faster, Stronger uses a DigiTech Talker vocoder. Given the DigiTech Talker was used extensively for Human After All, maybe it was one of the last songs recorded for Discovery? The DigiTech Talker wasnâ■■t mentioned in the May 2001 interview, despite its use on Discovery. Human After All notes (14 March 2005) # DigiTech Talker and DigiTech Synth Wah are all over the entire album. But, did they use a DigiTech Synth Wah, or DigiTech Bass Synth Wah? Theyâ■■re very similar pedals. The tremolo effect on The Brainwasher could have been done many ways. Maybe it was just an LFO modulating the amplitude on their Roland S-760 sampler? Maybe it was a guitar pedal? Itâ■■s an easy effect that can be achieved many different ways. Random Access Memories notes (17 May 2013) # In Lose Yourself To Dance, the â■■everybodyâ■■s dancing on the floorâ■■ vocals sound very crunchy and DigiTech Talker-like. The vocodeded vocals in Touch sound like a Sennheiser VSM201 switched to unvoiced, or using white noise as the vocoderâ■■s carrier. Instant Crush could be Auto-Tune or some other kind of harmoniser. It sounds like Instant Crush constains some Sennheiser VSM201 chord layers in places. Talk boxes # Daft Punkâ■■s vocal effects can be broadly split into three categories: Talk boxes, vocoders, and harmonisers. They all sound vaguely similar and robot-like, and you could be forgiven for confusing them, but theyâ■■re extremely different techniques and technologies. Talk boxes are relatively simple devices â■■ theyâ■■re a speaker in a sealed box with a small opening. One end of a hose is fitted to the opening, and the other end is placed into the performerâ■■s mouth, blasting noise towards their throat. The performer can pretend to speak, shaping and filtering the sound coming out of the tube with their vocal tract. A microphone is then needed to record the resulting sound. A keyboard or guitar is typically connected to the talk box unit as the sound source for the speaker. This lets the keyboard or guitar sound like itâ■■s singing. If youâ■■ve heard Chromeo, 2Pacâ■■s California Love, Peter Framptonâ■■s Do You Feel Like We Do, or Bon Joviâ■■s Livinâ■■ On A Prayer before, youâ■■ve heard a talk box. I can confirm firing loud sounds into your mouth while holding a tube with your teeth is a bit uncomfortable. In terms of vocal effects used by Daft Punk, I think talk box might be the least used and least interesting, in terms of hunting down the exact hardware used. Talk boxes are simple devices and typically all sound similar. The sound source and performance play a bigger role in the result than the hardware itself. Also, there arenâ■■t many talk boxes on the market. Daft Punk may have used a Heil Talk Box, a Rocktron Banshee, a home made talk box, or something else. The MXR M222 Talk Box is probably the best option if youâ■■re looking to buy a talk box today, because it has a built in amplifier. The MXR wasnâ■■t around when Around The World was created though, so thatâ■■s not the unit they used. Daft Punkâ■■s early albums extensively used a Roland Juno-106, so itâ■■s likely that was the sound source for the talk box used on Around The World. It sounds like a sawtooth wave with the filters open. Even though theyâ■■ve been around in a commercial form since the mid 70s, talk boxes arenâ■■t the first device to use human vocal tracts to create robotic sounds â■■ the Sonovox from 1939 takes that prize. Vocoders # Vocoders are a bit like an electronic version of a talk box. Vocoders take two audio inputs â■■ often a voice and a synth â■■ and combine them by filtering the synth with the voiceâ■■s frequency response. The filtering is usually done by splitting the signal into frequency bands. The volume of each voice band sets the volume of the repective synth band. More bands usually means a higher quality and more intelligible result. Iâ■■ve been calling the inputs â■■voiceâ■■ and â■■synthâ■■, but theyâ■■re often referred to as the modulator and carrier. The modulator filters the carrier. Vocoders can be analogue or digital. Good analogue vocoders are physically big and very expensive, due to their complexity, especially if they have lots of frequency bands. Theyâ■■re also a specialty effect, and therefore usually not mass produced. The peak for high-end analogue vocoders was the 1970s â■■ the EMS Vocoder 5000 was released in 1976, the Bode/Moog Vocoder in 1977, and the Sennheiser VSM201 in 1977. Itâ■■s hard to know exactly how many Sen

3. Show HN: Bracket – selfhosted tournament system

GitHub - evroon/bracket: Selfhosted tournament system Skip to content You signed in with another tab or window. Reload to refresh your session. You signed out in another tab or window. Reload to refresh your session. You switched accounts on another tab or window. Reload to refresh your session. Dismiss alert evroon / bracket Public Notifications You must be signed in to change notification settings Fork 70 Star 335 Selfhosted tournament system docs.bracketapp.nl License AGPL-3.0 license 335 stars 70 forks Branches Tags Activity Star Notifications You must be signed in to change notification settings evroon/bracket masterBranchesTagsGo to fileCodeFolders and filesNameNameLast commit messageLast commit dateLatest commit History920 Commits.github.github backendbackend docs docs frontendfrontend .gitignore.gitignore LICENSELICENSE README.mdREADME.md SECURITY.mdSECURITY.md codecov.ymlco decov.yml crowdin.ymlcrowdin.yml docker-compose.ymldocker-compose.yml process-compose-example.ymlprocess-compose-example.yml run.shrun.sh View all filesRepository files navigation Demo · Documentation · Quickstart · GitHub · Releases Tournament system meant to be easy to use. Bracket is written in async Python (with FastAPI) and Next.js as frontend using the Mantine library. It has the following features: Supports single elimination, round-robin and swiss formats. Build your tournament structure with multiple stages that can have multiple groups/brackets in them. Drag-and-drop matches to different courts or reschedule them to another start time. Various dashboard pages are available that can be presented to the public, customized with a logo. Create/update teams, and add players to teams. Create multiple clubs, with multiple tournaments per club. Swiss tournaments can be handled dynamically, with automatic scheduling of matches. Explore the Bracket docs ▶️ Live Demo A demo is available for free at https://www.bracketapp.nl/demo. The demo lasts for 30 minutes, after which your data will de deleted. Quickstart To quickly run bracket to see how it works, clone it and run docker compose up: git clone git@github.com:evroon/bracket.git cd bracket sudo docker compose up -d This will start the backend and frontend of Bracket, as well as a postgres instance. You should now be able to view bracket at http://localhost:3000. You can log in with the following credentials: Username: test@example.org Password: aeGhoe1ahng2Aezai0Dei6Aih6dieHoo. To insert dummy rows into the database, run: sudo docker exec bracket-backend pipenv run ./cli.py create-dev-db See also the quickstart docs. Usage Read the usage guide for how to organize a tournament in Bracket from start to finish. Configuration Read the configuration docs for how to configure Bracket. Bracket's backend is configured using .env files (prod.env for production, dev.env for development etc.). But you can also configure Bracket using environment variables directly, for example by specifying them in the docker-compose.yml. The frontend doesn't can be configured by environment variables as well, as well as .env files using Next.js' way of loading environment variables. Running Bracket in production Read the deployment docs for how to deploy Bracket and run it in production. Bracket can be run in Docker or by itself (using pipenv and yarn). Development setup Read the development docs for how to run Bracket for development. Prerequisites are yarn, postgresql and pipenv to run the frontend, database and backend. Translations Based on your browser settings, your language should be automatically detected and loaded. For now, there's no manual way of choosing a different language. Supported Languages To add/refine translations, Crowdin is used. See the docs for more information. More screenshots Help If you're having trouble getting Bracket up and running, or have a question about usage or configuration, feel free to ask. The best place to do this is by creating a Discussion. Supporting Bracket If you're using Bracket and would like to help support its development, that would be greatly appreciated! Several areas that we need a bit of help with at the moment are: ⭐ Star Bracket on GitHub 🌐 Translating: Help make Bracket available to non-native English speakers by adding your language (via crowdin) 📣 Spread the word by sharing Bracket to help new users discover it 🖥️ Submit a PR to add a new feature, fix a bug, extend/update the docs or something else See the contribution docs for more information on how to contribute Contributors Erik Vroon Null BachErik Danny Piper SevicheCC Nicolas Vanheuverzwijn IzStriker Raphael Le Goaller License Bracket is licensed under AGPL-v3.0. Please note that any contributions also fall under this license. See LICENSE About Selfhosted tournament system docs.bracketapp.nl Topics react python api docker json web yarn brackets reactjs nextjs postgresql selfhosted sports bracket tournaments tournament-manager tournament-bracket docusaurus fastapi mantine Resources Readme License AGPL-3.0 license Security policy Security policy Activity Stars 335 stars Watchers 5 watching Forks 70 forks Report repository Releases 37 v2.2.2 Latest Apr 4, 2025 + 36 releases Packages 0 Contributors 10 Languages Python 59.3% TypeScript 38.8% CSS 1.0% Dockerfile 0.3% JavaScript 0.3% Shell 0.2% Mako 0.1% You can’t perform that action at this time.

4. I'd rather read the prompt

I'd rather read the prompt I'd rather read the prompt Clayton Ramsey - 2025-05-03 When I grade students’ assignments, I sometimes see answers like this: Utilizing Euler angles for rotation representation could have the following possible downsides: Gimbal lock: In certain positions, orientations can reach a singularity, which prevents them from continuously rotating without a sudden change in the coordinate values. Numeric instability: Using Euler angles could cause numeric computations to be less precise, which can add up and produce inaccuracies if used often. Non-unique coordinates: Another downside of Euler angles is that some rotations do not have a unique representation in Euler angles, particularly at singularities. The downsides of Euler angles make them difficult to utilize in robotics. It’s important to note that very few implementations employ Euler angles for robotics. Instead, one could use rotation matrices or quaternions to facilitate more efficient rotation representation. [Not a student’s real answer, but my handmade synthesis of the style and content of many answers] You only have to read one or two of these answers to know exactly what’s up: the students just copy-pasted the output from a large language model, most likely ChatGPT. They are invariably verbose, interminably waffly, and insipidly fixated on the bullet-points-with-bold style. The prose rarely surpasses the sixth-grade book report, constantly repeating the prompt, presumably to prove that they’re staying on topic. As an instructor, I am always saddened to read this. The ChatGPT rhetorical style is distinctive enough that I can catch it, but not so distinctive to be worth passing along to an honor council. Even if I did, I’m not sure the marginal gains in the integrity of the class would be worth the hours spent litigating the issue. I write this article as a plea to everyone: not just my students, but the blog posters and Reddit commenters and weak-accept paper authors and Reviewer 2. Don’t let a computer write for you! I say this not for reasons of intellectual honesty, or for the spirit of fairness. I say this because I believe that your original thoughts are far more interesting, meaningful, and valuable than whatever a large language model can transform them into. For the rest of this piece, I’ll briefly examine some guesses as to why people write with large language models so often, and argue that there’s no good reason to use one for creative expression. Why do people do this? I’m not much of a generative-model user myself, but I know many people who heavily rely upon them. From my own experience, I see a few reasons why people use such models to speak for them. It doesn’t matter. I think this belief is most common in classroom settings. A typical belief among students is that classes are a series of hurdles to be overcome; at the end of this obstacle course, they shall receive a degree as testament to their completion of these assignments. I think this is also the source of increasing language model use in in paper reviews. Many researchers consider reviewing ancillary to their already-burdensome jobs; some feel they cannot spare time to write a good review and so pass the work along to a language model. The model produces better work. Some of my peers believe that large language models produce strictly better writing than they could produce on their own. Anecdotally, this phenomenon seems more common among English-as-a-second-language speakers. I also see it a lot with first-time programmers, for whom programming is a set of mysterious incantations to be memorized and recited. I think this is also the cause of language model use in some forms of academic writing: it differs from the prior case with paper reviews in that, presumably, the authors believe that their paper matters, but don’t believe they can produce sufficient writing. There’s skin in the game. This last cause is least common among individuals, but probably accounts for the overwhelming majority of language pollution on the Internet. Examples of skin-in-the-game writing include astroturfing, customer service chatbots, and the rambling prologues found in online baking recipes. This writing is never meant to be read by a human and does not carry any authorial intent at all. For this essay, I’m primarily interested in the motivations for private individuals, so I’ll avoid discussing this much; however, I have included it for sake of completeness. Why do we write, anyway? I believe that the main reason a human should write is to communicate original thoughts. To be clear, I don’t believe that these thoughts need to be special or academic. Your vacation, your dog, and your favorite color are all fair game. However, these thoughts should be yours: there’s no point in wasting ink to communicate someone else’s thoughts. In that sense, using a language model to write is worse than plagiarism. When copying another person’s words, one doesn’t communicate their own original thoughts, but at least they are communicating a human’s thoughts. A language model, by construction, has no original thoughts of its own; publishing its output is a pointless exercise. Returning to our reasons for using a language model, we can now examine them once more with this definition in mind. If it’s not worth doing, it’s not worth doing well The model output in the doesn’t-matter category falls under two classes to me: the stuff that actually doesn’t matter and the stuff that actually does matter. I’ll start with the things that don’t matter. When someone comments under a Reddit post with a computer-generated summary of the original text, I honestly believe that everyone in the world would be better off had they not done so. Either the article is so vapid that a summary provides all of its value, in which case, it does not merit the engagement of a comment, or it demands a real reading by a real human for comprehension, in which case the summary is pointless. In essence, writing such a comment wastes everyone’s time. This is the case for all of the disposable uses of a model. Meanwhile, there are uses which seem disposable at a surface-level and which in practice are not so disposable (the actually-does-matter category). I should hope that the purpose of a class writing exercise is not to create an artifact of text but force the student to think; a language model produces the former, not the latter. For paper reviewers, it’s worse: a half-assed review will produce little more than make-work for the original authors and tell the editor nothing they didn’t already know. If it’s worth doing, it’s worth doing badly I’ll now cover the opposite case: my peers who see generative models as superior to their own output. I see this most often in professional communication, typically to produce fluff or fix the tone of their original prompts. Every single time, the model obscures the original meaning and adds layers of superfluous nonsense to even the simplest of ideas. If you’re lucky, it at least won’t be wrong, but most often the model will completely fabricate critical details of the original writing and produce something completely incomprehensible. No matter how bad any original human’s writing is, I can (hopefully?) trust that they have some kind of internal understanding to share; with a language model, there is no such luck. I have a little more sympathy for programmers, but the long-term results are more insidious. You might recall Peter Naur’s Programming as Theory Building: writing a sufficiently complex program requires not only the artifact of code (that is, the program source), but a theory of the program, in which an individual must fully understand the logical structure behind the code. Vibe coding; that is, writing programs almost exclusively by language-model generation; produces an artifact with no theory behind it. The result is simple: with no theory, the produced code is practically useless. In Naur’s terms, such a pro

5. AI Meets WinDBG

The Future of Crash Analysis: AI Meets WinDBGThe Future of Crash Analysis: AI Meets WinDBGPosted on 2025-05-04Old Meets New: Bringing Crash Analysis into 2025Let's face it - while the rest of software development has evolved at warp speed, crash dump analysis feels like it's been preserved in digital amber for decades. We've got self-driving cars and pocket-sized supercomputers, yet here we are, still pecking away at command prompts like it's the dawn of the internet. Why is debugging the only area where we cling to tools that look like they belong in a computer history museum?Picture this: You, a professional software engineer in 2025, hunched over a terminal, manually typing arcane commands like !analyze -v and .ecxr, squinting at hexadecimal memory addresses, and mentally translating stack traces. All while your friends in other industries are delegating their work to AI assistants that can write entire documents, create art, or automate complex workflows.Something's wrong with this picture, right?What if I told you we can throw that ancient workflow into the dustbin of computing history? That's exactly what I've done. And I'm not talking about slightly better syntax highlighting or prettier UI for WinDBG. I'm talking about a fundamental transformation where you simply have a conversation with your debugger.When Inspiration StrikesDuring a debugging session at work, I had one of those lightning bolt moments. What if - and stick with me here - we could apply the same AI-assisted "vibe coding" approach to crash dump analysis?Picture this: instead of manually slogging through memory dumps and command outputs, you simply ask, "Hey, why did this application crash?" and get an intelligent, contextual answer that actually helps you solve the problem.The idea was too compelling not to pursue. So I built it.See It In Action: AI-Powered Crash AnalysisBefore diving into the technical details, let me show you what this looks like in practice. I have prepared a demo application to showcase two different use cases:Video 1: Crash Analysis and Automated BugfixIn this video, I show how Copilot can analyze a crash dump, identify the bug and auto-fix the issue.Your browser does not support the video tag.As you can see in the video, instead of manually running WinDBG commands and interpreting the cryptic output, I'm having a natural conversation with GitHub Copilot. The AI quickly identifies that the application crashed, explains which specific conditions led to the crash, and suggests a fix.Video 2: Automated Crash Dump Analysis of multiple crash dump filesThis video demonstrates a different capability: analyzing multiple crash dump files at once. It shows how the tool can quickly identify which dumps belong to your application and which don't.Your browser does not support the video tag.Worth noting, it takes just a few seconds until you get your first useful answer. I've played around with this for many hours and let me tell you one thing: You can really go deep. If you ask the right questions, the AI runs WinDBG/CDB commands that I haven't seen in all these years of debugging, and that is simply amazing.How can this help the industry?I believe this is one of the really good examples of how AI can boost productivity. Analyzing crash dumps is a very tedious task. It begins with quickly checking and identifying whether crashes are the same or different, and often requires very advanced knowledge when a crash is challenging - really challenging.Copilot can help here tremendously; it knows how to:Interpret assembly code (without you having to remember what EAX stands for)Check memory contents (so you don't have to count hex bytes on your fingers)Traverse structures with symbols (goodbye to manual pointer arithmetic!)And so much moreThis is a game changer - not just for engineers, but also for support, QA, and everyone involved with crash dumps. It's like going from hunting with a stone spear to using a guided missile.How did I build this?If you've ever worked with WinDBG, you know the drill: cryptic commands, obscure syntax, and endless scrolling through memory addresses and stack traces that make your eyes glaze over. It's the kind of specialized knowledge that takes years to master and feels like speaking an alien language even when you do.The trick here is connecting WinDBG with AI. To do that, you first need to programmatically control a debugging session, right? There are plenty of options on how to do this. I prefer to keep things simple, so I have chosen CDB, which is Microsoft's Console Debugger. It operates on standard input and output, and that's so much more fun to deal with than setting up COM APIs or similar approaches.The second part is "connecting with AI." That's where Model Context Protocol Servers come into the game.Understanding Model Context Protocol ServersMCP is an open standard developed by Anthropic, released in November 2024. This protocol allows AI models to interact with external tools and data sources - think of it as giving AI assistants "hands" to work with other software. It defines a way for AI assistants to discover, access, and use tools through a consistent interface. In essence, it's what allows GitHub Copilot to "talk" to external programs like WinDBG.An MCP server acts as the intermediary between the AI model and the tool. It:Registers available tools with the clientHandles requests from AI models to use these toolsExecutes the tool operations and returns resultsMaintains context across interactionsThis architecture means that any tool can be made available to AI models if someone builds an MCP server for it. And that's exactly what I did for WinDBG (CDB).Why MCP Instead of LanguageModelTool API?The LanguageModelTool API might eventually be a better fit for this specific use-case. Creating a Visual Studio Extension that "just works" out of the box would potentially simplify the integration process significantly.However, using MCP directly offers several notable advantages. It works with any AI model, not just limiting itself to Copilot. The server can be used outside VS Code, functioning with various other tools. New features can be easily added without necessitating changes to the core integration. Moreover, it remains platform-independent, avoiding lock-in to any single company's implementation.The MCP-WinDBG ProjectI've implemented a Model Context Protocol server that wraps WinDBG/CDB and exposes its capabilities to AI models within VS Code. Better yet, I've made it open source so everyone can experience this new workflow.The project, called mcp-windbg, creates a seamless bridge between VS Code, GitHub Copilot, and the powerful analysis capabilities of WinDBG.The actual "hard part" was implementing the CDB (Command-Line WinDBG) interaction layer. And by "hard", I mean vibe-coding with two coffees on a Saturday morning, where I spent more time being annoyed by pyTest failures than actual coding difficulties. The core implementation came together surprisingly quickly!The rest is primarily wrapper code that implements the Model Context Protocol specifications. Now that I've established and defined the core WinDBG interaction logic, I'm considering refactoring the project to TypeScript. This would enable me to create both an MCP Server in TypeScript and a dedicated Visual Studio Extension, with both implementations leveraging the same underlying CDB interaction layer.What Does This Mean In Practice?Let me walk you through what this enables:Natural language crash analysis: "Why is this application crashing with an access violation at this address?" (Instead of: "What the $%#@ is this heap corruption!?")Contextual debugging: "Show me the stack trace for thread 5 and explain what each function is doing based on the symbols." (Instead of staring at call stacks like they're ancient hieroglyphics)Root cause identification: "What's causing this null pointer dereference and where should I look in the code to fix it?" (Instead of playing d