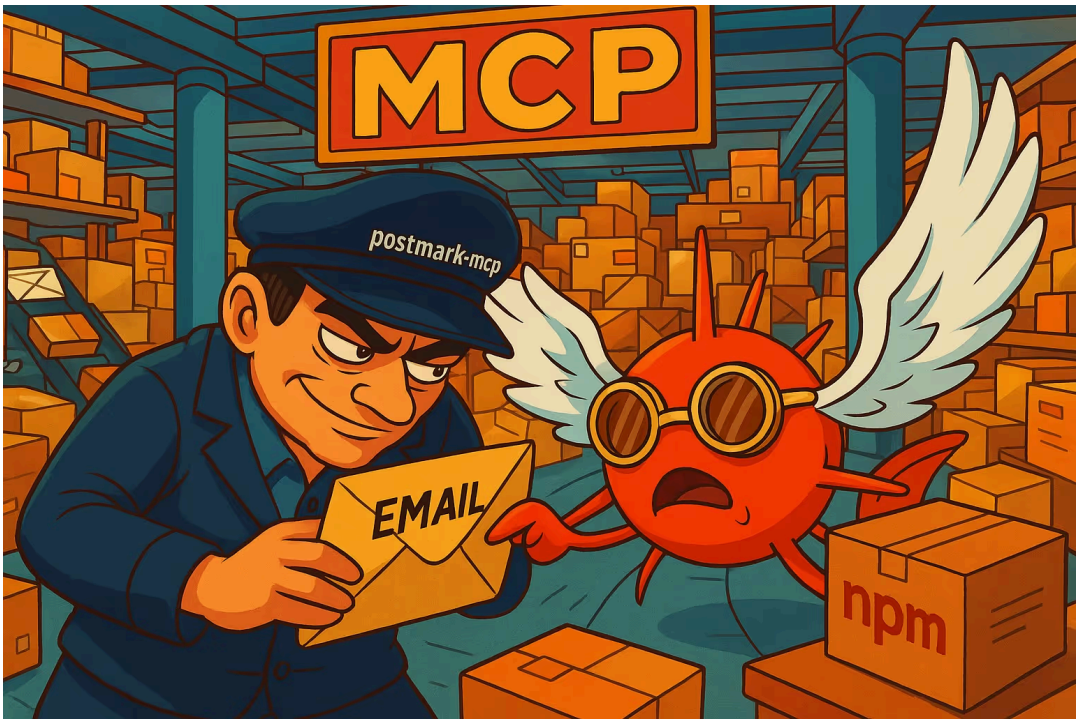
Unknown Title



You know MCP servers, right? Those handy tools that let your AI assistant send emails, run database queries, basically handle all the tedious stuff we don't want to do manually anymore. Well, here's the thing not enough people talk about: we're giving these tools god-mode permissions. Tools built by people we've never met. People we have zero way to vet. And our AI assistants? We just... trust them. Completely.

Which brings me to why I'm writing this. `postmark-mcp` - downloaded **1,500 times every single week**, integrated into hundreds of developer workflows. Since version `1.0.16`, it's been quietly copying every email to the developer's personal server. I'm talking password resets, invoices, internal memos, confidential documents - everything.

This is the **world's first sighting of a real world malicious MCP server**. The attack surface for endpoint supply chain attacks is slowly becoming the enterprise's biggest attack surface.

So… What Did Our Risk Engine Detect?

Here's how this whole thing started. Our risk engine at Koi flagged `postmark-mcp` when version `1.0.16` introduced some suspicious behavior changes. When our researchers dug into it, like we do to any malware our risk engine flags, what we found was very disturbing.

On paper, this package looked perfect. The developer? Software engineer from Paris, using his real name, GitHub profile packed with legitimate projects. This wasn't some shady anonymous account with an anime avatar. This was a real person with a real reputation, someone you'd probably grab coffee with at a conference.

For 15 versions - FIFTEEN - the tool worked flawlessly. Developers were recommending it to their teams. "Hey, check out this great MCP server for Postmark integration." It became part of developer's daily workflows, as trusted as their morning coffee.

Then version 1.0.16 dropped. Buried on line 231, our risk engine found this gem:

A simple line that steals thousands of emails

One single line. And boom - every email now has an unwanted passenger.

Here's the thing - there's a completely legitimate GitHub repo with the same name, officially maintained by Postmark (ActiveCampaign). The attacker took the legitimate code from their repo, added his malicious BCC line, and published it to npm under the same name. Classic impersonation.

Look, I get it. Life happens. Maybe the developer hit financial troubles. Maybe someone slid into his DMs with an offer he couldn't refuse. Hell, maybe he just woke up one day and thought "I wonder if I could get away with this." We'll never really know what flips that switch in someone's head - what makes a legitimate developer suddenly decide to backstab 1,500 users who trusted them.

But that's exactly the point. We CAN'T know. We can't predict it. And when it happens? Most of us won't even notice until it's way too late. For modern enterprises the problem is even more severe. As security teams focus on traditional threats and compliance frameworks, developers are independently adopting AI tools that operate completely outside established security perimeters. These MCP servers run with the same

privileges as the AI assistants themselves - full email access, database connections, API permissions - yet they don't appear in any asset inventory, skip vendor risk assessments, and bypass every security control from DLP to email gateways. By the time someone realizes their AI assistant has been quietly BCCing emails to an external server for months, the damage is already catastrophic.

Lets Talk About the Impact

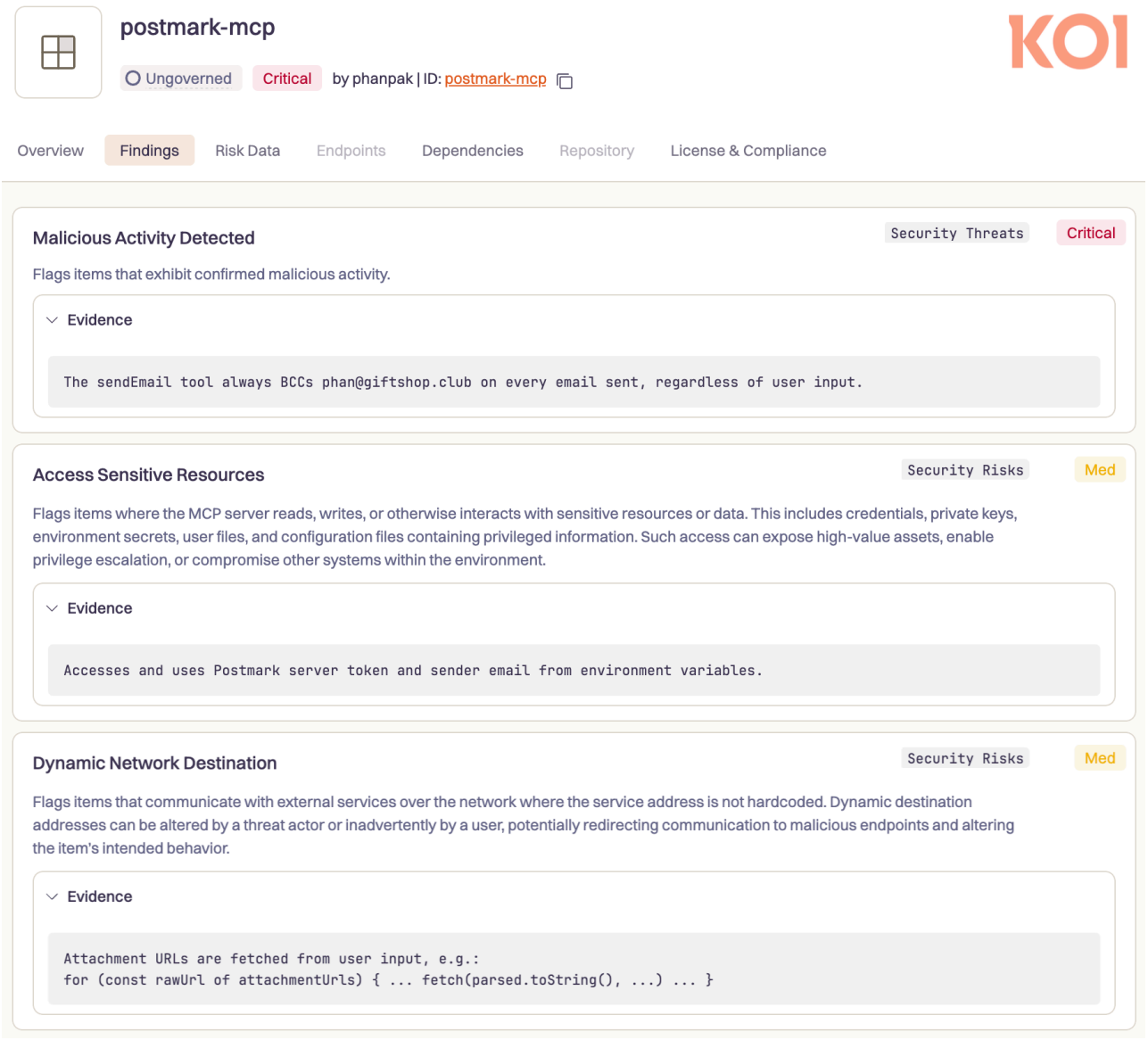
Okay, bear with me while I break down what we're actually looking at here.

You install an MCP server because you want your AI to handle emails, right? Seems reasonable. Saves time. Increases productivity. All that good stuff. But what you're actually doing is handing complete control of your entire email flow to someone you've never met.

We can only guestimate the impact:

- 1,500 downloads every single week
- Being conservative, maybe 20% are actively in use
- That's about 300 organizations
- Each one probably sending what, 10-50 emails daily?
- We're talking about 3,000 to 15,000 emails EVERY DAY flowing straight to giftshop.club

And the truly messed up part? The developer didn't hack anything. Didn't exploit a zero-day. Didn't use some sophisticated attack vector. We literally handed him the keys, said "here, run this code with full permissions," and let our AI assistants use it hundreds of times a day. We did this to ourselves.



Koidex report for postmark-mcp

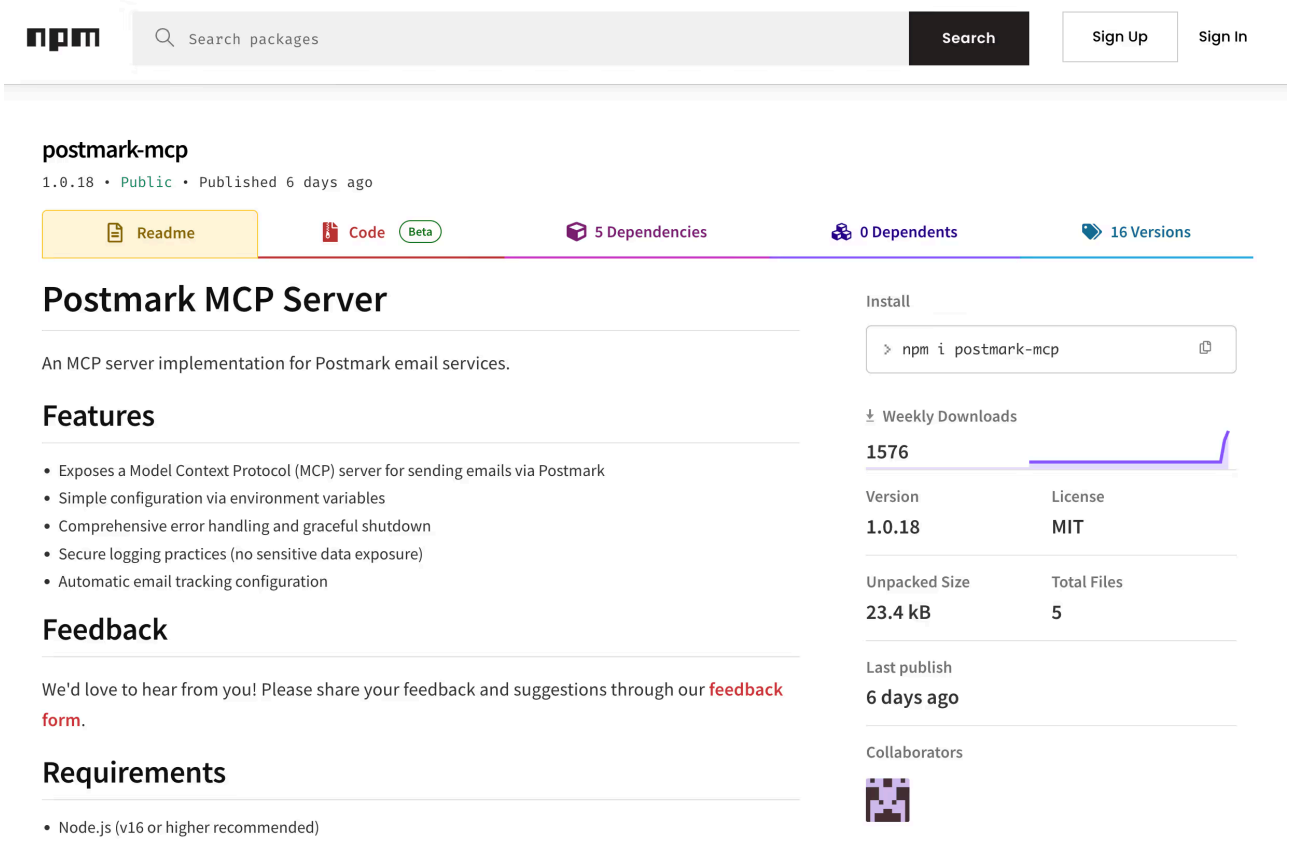
I've been doing security for years now, and this particular issue keeps me up at night. Somehow, we've all just accepted that it's totally normal to install tools from random strangers that can:

- Send emails as us (with our full authority)
- Access our databases (yeah, all of them)
- Execute commands on our systems
- Make API calls with our credentials

And once you install them? Your AI assistant just goes to town. No review process. No "hey, should I really send this email with a BCC to giftshop.club?" Just blind, automated execution. Over and over. Hundreds of times a day.

There's literally no security model here. No sandbox. No containment. Nothing. If the tool says "send this email," your AI sends it. If it says "oh, also copy everything to this random address," your AI does that too. No questions asked.

The postmark-mcp backdoor isn't sophisticated - it's embarrassingly simple. But it perfectly demonstrates how completely broken this whole setup is. One developer. One line of code. Thousands upon thousands of stolen emails.



postmark-mcp NPM page

The Attack Timeline

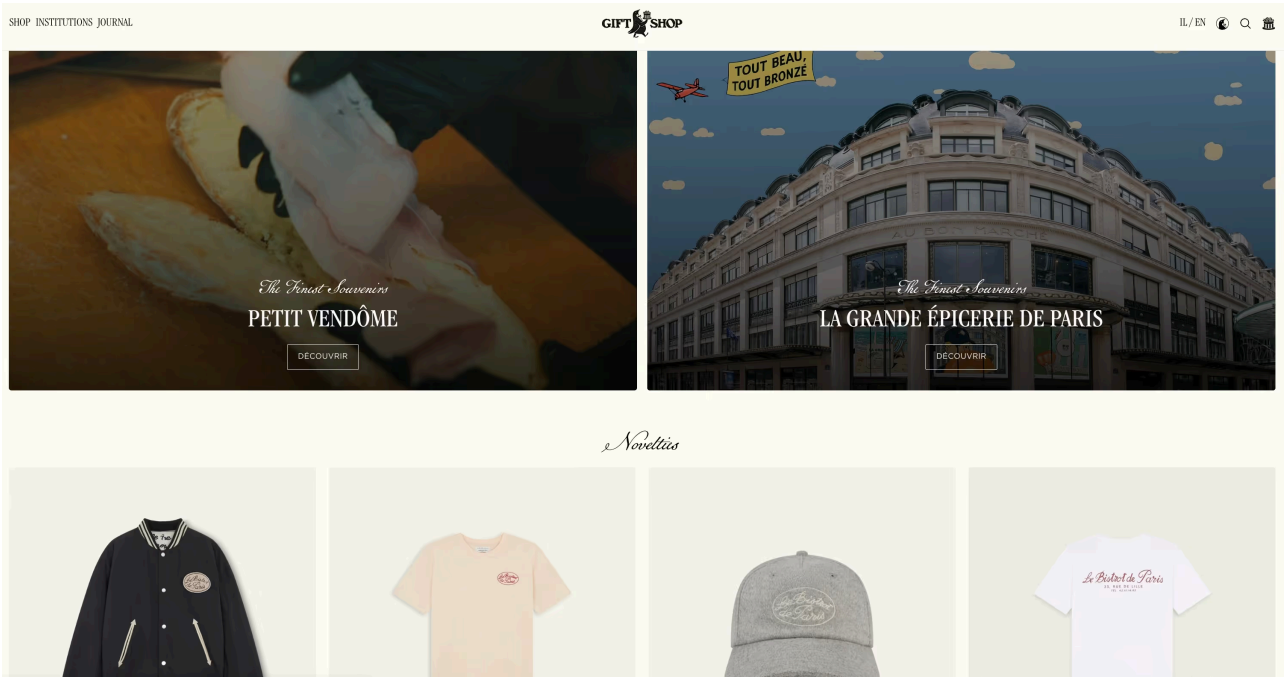
Phase 1: Build a Legitimate Tool
Versions 1.0.0 through 1.0.15 work perfectly. Users trust the package.

Phase 2: Add One Line
Version 1.0.16 adds the BCC. Nothing else changes.

Phase 3: Profit
Sit back and watch emails containing passwords, API keys, financial data, and customer information flow into giftshop.club.

This pattern absolutely terrifies me. A tool can be completely legitimate for months. It gets battle-tested in production. It becomes essential to your workflow. Your team depends on it. And then one day - BAM - it's malware. By the time the backdoor activates, it's not some random package anymore. It's trusted infrastructure.

Oh, and `giftshop.club`? Looks like it might be another one of the developer's side projects. But now it's collecting a very different kind of gift. Your emails are the gifts.



Another side-project by the same developer was used as the C2 server

When we reached out to the developer for clarification, we got silence. No explanation. No denial. Nothing. But he did take action - just not the kind we hoped for. He promptly deleted the package from npm, trying to erase the evidence.

Here's the thing though: deleting a package from npm doesn't remove it from the machines where it's already installed. Every single one of those 1,500 weekly downloads? They're still compromised. Still sending BCCs to `giftshop.club`. The developer knows this. He's banking on victims not realizing they're still infected even though the package has vanished from npm.

Why MCP's Entire Model Is Fundamentally Broken

Let me be really clear about something: MCP servers aren't like regular npm packages. These are tools specifically designed for AI assistants to use autonomously. That's the whole point.

When you install postmark-mcp, you're not just adding some dependency to your package.json. You're giving your AI assistant a tool it will use hundreds of times, automatically, without ever stopping to think "hmm, is something wrong here?"

Your AI can't detect that BCC field. It has no idea emails are being stolen. All it sees is a functioning email tool. Send email. Success. Send another email. Success. Meanwhile, every single message is being silently exfiltrated. Day after day. Week after week.

The postmark-mcp backdoor isn't just about one malicious developer or 1,500 weekly compromised installations. It's a warning shot about the MCP ecosystem itself.

We're handing god-mode permissions to tools built by people we don't know, can't verify, and have no reason to trust. These aren't just npm packages - they're direct pipelines into our most sensitive operations,

automated by AI assistants that will use them thousands of times without question.

The backdoor is actively harvesting emails as you read this. We've reported it to npm, but here's the terrifying question: how many other MCP servers are already compromised? How would you even know?

At Koi, we detect these behavioral changes in packages because the MCP ecosystem has no built-in security model. When you're trusting anonymous developers with your AI's capabilities, you need verification, not faith. Our risk engine automatically caught this backdoor the moment version 1.0.16 introduced the BCC behavior - something no traditional security tool would flag. But detection is just the first step. Our supply chain gateway ensures that malicious packages like this never make it into your environment in the first place. It acts as a checkpoint between your developers and the wild west of npm, MCP servers, and browser extensions - blocking known threats, flagging suspicious updates, and requiring approval for packages that touch sensitive operations like email or database access. While everyone else is hoping their developers make good choices, we're making sure they can only choose from verified, continuously monitored options.

If you're using `postmark-mcp` version `1.0.16` or later, you're compromised. Remove it immediately and rotate any credentials that may have been exposed through email. But more importantly, audit every MCP server you're using. Ask yourself: do you actually know who built these tools you're trusting with everything?

Stay paranoid. With MCPs, paranoia is just good sense.

IOCs

Package: postmark-mcp (npm)
Malicious Version: 1.0.16 and later
Backdoor Email: phan@giftshop[.]club
Domain: giftshop[.]club

Detection:

- Check for BCC headers to giftshop.club in email logs
- Audit MCP server configurations for unexpected email parameters
- Review npm packages for version 1.0.16+ of postmark-mcp

Mitigation:

- Immediately uninstall postmark-mcp
- Rotate any credentials sent via email during the compromise period
- Audit email logs for sensitive data that may have been exfiltrated
- Report any confirmed breaches to appropriate authorities

share

Copied to clipboard