Slide 3

I'm the founder and CEO of Packetware, a cybersecurity-focused content delivery cloud platform. I started the company because I was building many technologies on the Internet, but had issues with cyberattacks. Even though I was competent in the world of web servers and platforms such as Cloudflare, I couldn't adequately protect my products. Other developers were having the same issue.

So, I reached out to teams and CEOs. I applied for internships with the goal of sharing what I had learned. I had correspondence with Cloudflare CEO Matthew Prince, but he was disinterested in solving the issues for small developers facing serious cyberattacks.

There was one option left. Do it myself. I have led Packetware's engineering from the ground up, from custom Rust servers, to eBPF, to client-side JavaScript. Everything has been designed to make user experiences more satisfying and rewarding with a new secure-by-default philosophy – by taking abstraction out of the product.

Part of that central goal is to stop asking humans if they are humans. The Web exists for us. We shouldn't be the ones clicking the traffic lights, or being asked for additional security checks.

Thus, according to first principles, a visual solution is a logic fail. The future of the web can't rely on human interaction. We can have it operate otherwise.

So, I ambitiously attempted to retry the solution to bots. To influence change in a better direction.

This is not an end solution. With the rise of AI, the environment will dynamically change. But this provides lessons for today.

Slide 4

We reverse proxy HTTP and HTTPS content from origins to clients, saving temporary copies of content globally to speed up delivery, and making sure users are legitimate along the way.

When we're dubious of a request, whether it be from server-side fingerprints or current traffic flowing into the domain from a number of heuristics, we serve them with a JavaScript "challenge" to complete before allowing them access to the resources.

If the request seems fine at first glance, we inject the JavaScript challenge at the end of response streams to verify their authenticity in the background. That way, we can allow them access to privileged APIs without interruption.

Slide 5

What makes a browser like Chrome, Safari, or Firefox legitimate?

- Easily differentiable from simple bots, but CAPTCHAs don't exist to solve that problem
- Emulated browsers
    - They have JavaScript windows that are emulated to be real browsers, but they are "headless" (they don't have a real display and are controlled)
    - Have similar characteristics, but fundamental differences
- They are run in different environments. They have different error stacks, they have different memory characteristics, they have hidden control tools embedded, they have different video cards, they handle notifications differently, they have different sound stacks
- They have interior APIs that make the DOM and other browser elements easier to control
    - Special memory APIs

Slide 6

- Recursive functions take up "stack size" each time they are iterated upon
- Chrome restricts functions to have a call depth of 8,950
    - Discovered when developing another check
- Headless browsers (like Selenium and Puppeteer) that pretend to be Chrome don't have the same memory restriction
    - They have a deeper call depth so the controller has more flexibility that wouldn't normally be allowed in a browser

Slide 7 (possible TODO: change the check on this slide, it may be too basic)

This is a more specific check than the other, intentionally throwing an error (that doesn't actually print to the console). It then reads the error stack, checking if it includes Puppeteer.

- Puppeteer is the most commonly used headless browser
- Found whilst playing around with Puppeteer and noticed backtraces showed the emulator
- Cannot be removed from the error stack without serious forking

But there are problems

- Our challenge bundles about 30 of checks similar to these and more complex, including invisibly painting to the DOM.
- False positives → a browser on different platforms and different conditions looks differently.
    - These checks cannot be applied statically. If a couple checks fails but the others work and there's no indications of foul play, we have to stay dynamic and respond to the conditions in the moment. Our systems automatically adjust the threshold for which checks work and don't work. Principally, sticking to the notion that users don't notice the best cybersecurity platforms.

    - It's still effective against every emulated browser tested
- But at the very minimum, the challenge provides a requirement for a proof of work baseline that's able to adjust with incoming traffic.

Slide 8

- This code is being ran on the client device, so we're principally not in a secure environment
- Script communicates with an edge server to validate challenges and share check results. While data sent and received is encrypted with AES-128 GCM, the keys lie within the code
    - Keys being exposed is a critical error where users can be issued cookies without going through the challenge at all by constructing false information

- Local solution → obfuscation
    - JavaScript's hacky syntax allows us to "obfuscate" the code by messing around with it, partially hiding the checks, but mostly hiding the keys
        - Doing it myself → starting off by hiding the keys around the script, and performing character-code mutations and displaying them differently
    - You might be saying to yourself, "there is no security through obscurity," so how else can we keep the challenge legitimate?

- Changing the characteristics of the game
    - Currently, we're thinking about keeping this data secure on a static lifetime. That's principally impossible.

Slide 9

Securing the keys on an unlimited timeframe not only is not very possible, but also conflicts with other priorities, such as the program speed, debuggability, and size as we think about delivering it to devices.

So why not change the timeframe?

- Having the central server rotate challenges and challenge encryption keys every 100 seconds.
    - Creating an obfuscator with inspiration from npm package javascript-obfuscator
        - Changing function and variable names, moving numbers to functions, control flow flattening
        - Most importantly, keeping it completely random
            - There's a new challenge every 100 seconds to solve from the ground up, with new keys that are hidden in a unique way

Slide 10

- Explain slide, briefly reiterating upon each point, then:
    - My part in creating this
        - While I'm the one that put the script together from end to finish, I took lots of inspiration along the way. No great thing is built alone. For example, a few open source developers were willing to play around with me, trying different browser emulation strategies and different headless configurations, helping me develop challenges along the way and providing how they solved some of my issues. An advisor interested in the company's mission and I created the initially obfuscated script, and he taught me how best to hide the keys. I'm incredibly grateful for all the guidance I've received!
    - This is not a perfect solution. But I believe it is the first solution in the right direction. Humans should come first
    - This is just a part of the issues I'm trying to solve creatively at Packetware.

Slide 11

- Neuralink has one of the most ambitious goals that has high potential for society
    - I want to be part of and support a team that has high aspirations
    - Brain health is so important → I recently had a concussion and learning how to recover taught me a lot about how important and undervalued this sometimes can be

- I'm a good fit for this environment
    - I'm willing to go after the hard problems and appreciate having a bias towards action and learning along the way
    - I understand that goals can be highly ambitious and not have perfect solutions
        - Learning how to dynamically adapt and staying curious and open-minded after the first iteration doesn't work
            - Having a consistent "beginner's mind" is an important part of that for me
    - Playing the infinite game → I'm getting better day by day and it's our job as humans to improve the world
        - Being on teams where we push each other to improve the product and ourselves

Thank you so much for your time. Hope to speak soon!