EPISODE 1142

[INTRODUCTION]

[00:00:00] JM: A private network connects servers, computers and cloud instances. These networked objects are often separated by firewalls and subnets that create latency and complication.

David Crawshaw is the CTO of Tailscale, a company that works to make private networks easier to build and simpler to configure and maintain. David joins the show to talk about private networks and the implementation of Tailscale.

[INTERVIEW]

[00:00:31] JM: David, welcome to the show.

[00:00:33] DC: Thank you. It's nice to be here.

[00:00:35] JM: Today we're talking about private networks. Simple question; what is a private network?

[00:00:41] DC: Simple question. I don't think that's a simple question. That's a good question. What is a private network? So you'd think as someone who builds a VPN, which is a virtual private network, I would have a really good answer. But I suspect it's actually more complicated than that. There may not actually be a right answer. So maybe we could work from example here.

So consider the network in my house, for example. I have on a pretty standard ISP that you get in New York City and there's a router in the closet, and it's connected to a Wi-Fi access point, and the router gets a public IP address from the ISP and connects it to a NAT server and there's an internal DHCP server handing out private IP addresses to all the devices on my home network. And then it is connecting all of those devices to the Internet through the public IP address.

And so I would say that that's a private network in my house and it's connected to other networks through the Internet. The interconnection of networks is where the name Internet comes from. So I suppose that's a private network. But it's not entirely clear to me that I could take that and generalize it to a definition of private network. Because if we went to a café, if you remember the before times when you could go to cafés, and you took out your phone and you connected it to the cafés Internet Wi-Fi access point and got on the Internet that way, what's actually going on there is they have an ISP much like the one in my home. And there's router from that ISP, which has given them a public IP address. And then they have that router handing out private IP addresses on the café Wi-Fi. And then they have a NAT system for mapping those private IP addresses on to the public IP address in the same way. So the café has precisely the same kind of network I have in my house, but it's accessible to anyone who walks into the café and just connects to it. So I don't know if you could just call that a private network. What do you think?

[00:02:51] JM: I don't know. I mean, you're a company that builds private networks, or make private networks easier. So it sounds like the broad definition for private networks means you have a lot of work in front of you.

[00:03:05] DC: Yeah, I really does. The reason why think this is such a good question is that early on in our company when we had the beginnings of this product, I was sitting around thinking what should we call it. We said, "Should we call it a VPN?" And we argued about this a lot, because it's not clear that a VPN correctly explains what the product is to people.

So the problem here is that VPN is a very commonly used term that means at least two different things depending who you are. So probably the most common use of the term VPN are these privacy-related products that consumers download and install on their computer or their phone. There're lots of companies that provide these. And what these products effectively do is they take all of the Internet traffic that you try and send and receive on your device and proxy it through a server that the company runs at some other location.

And so back when these products were new, the primary use of them was how do I get to Netflix if it's geo-locked and I can't get in that country? So that's what a VPN was. Our product doesn't

even do that. So this is immediate confusion if we call it a VPN of like, “Oh, look. We’re a VPN. So I can watch Netflix.” No. No. You can’t. So that's one sort of big source of confusion. And that confusion existed before we were around, because companies with traditional corporate networks who have employees on laptops who want to access resources on the corporate network and store a piece of VPN software. These have been around for decades. And they use that to connect their laptop back to their company office. And that doesn't help with geo-block Netflix either and is a completely different thing.

So there’s at least two meanings of the term VPN in terms of products that people typically use. And where do we fit into that category and how do you tease apart those two notions is tricky. And I think your question, “What is a private network?” is a really good one because that's at the heart of the problem. If you take apart the up term virtual private network, I think we can all agree on what network means. We can get into the weeds on that one if you want. We can stop talking about fast interconnects on supercomputers and are those networks locked? It’s a bit messy. Is the interconnect between the sockets of CPUs on the motherboard a network? Well, that gets a bit complicated too.

But mostly, we agree it’s Ethernet, or Fiber, or Wi-Fi for connecting different computers using IP addresses. So we can agree on two of the three terms in the expression VPN. It's the third one that’s tricky, private. What does that mean? It means different things to different people. And I think there’s a couple different ways we could go after that depending how you want to explore it. We could talk about it historically, or we could look at –

[00:05:52] JM: Well, maybe we should just talk, get into Tailscale. So you ran a company called Tailscale, which builds private networks. Explain what you do.

[00:06:02] DC: Okay. So Tailscale builds a mesh VPN. And so the basic idea is you install Tailscale on two devices. And each of these devices plugs in using a standard identity access management system. So for example, you login with your Gmail address. And each machine gets an IP address, and then they can talk to each other. You can open a terminal on one device and ping the IP address on the other. You could write a little web server and make it accessible to the other device. You can install Tailscale on a third machine, and then it can see both the other machines you added. And no one else can see these IP addresses. They’re just shared

on your little virtual network.

So this is very similar to the traditional corporate VPN product, which comes as a server that you install in your office and then it comes as a client that you install on your laptop, for example, and then you can usually laptop to access your corporate network through the server you created. The significant technical difference here is that it's a mesh VPN, which means that the client is also the server software. Any two clients can connect to each other wherever they are. That is at the heart what we build.

[00:07:24] JM: So why is a secure network useful for the average enterprise company?

[00:07:31] DC: There's a historical answer to this. And there's the sort of going forward modern network answer to this. So the historical reason that's useful is that companies had corporate networks, and on these corporate networks, they had sensitive resource. And the only way to get to these sensitive resources was through the corporate network. And this was problematic when you weren't in your office and you weren't on the corporate network, but you need an access to these sensitive resources.

So VPNs were used to extend the reach of employees to these sensitive resources. It's often builds a security product. In some ways, VPNs are the opposite of a security product, because they expand the set of places and people who can access sensitive resource. So they're very much the opposite of security. That's the traditional explanation for why VPNs existed.

Tailscale is trying to create a new reason to use VPNs, which is – And this gets a little tricky to explain. But we are assigning identity to IP address. When you login to a Tailscale client with your Gmail address, we generate cryptographic key pair on that machine and transmit the public key to the other devices in your network. And the other devices in the network know that any traffic coming from that public key is the person who logged in with that Gmail address. And this gives you a new substrate for writing software. You can write software and see that packets are coming from a particular Tailscale IP address and you can know who is on the other end of that. And so this is typically security that is achieved at high-levels like in a web system. You create a TLS tunnel and then you have some way of demonstrating who the person is on the other end, either via some kind of OAuth 2 flow or some kind of identity system, like that ties together who

that tunnel is.

We move this down to the IP layer in the hopes that existing software that expects IP addresses to provide some sense of security, but also new software that wants a way to provide my identity system can be written without having to interface with all of these very complicated authentication mechanism. So we're hoping to build something new there.

[00:09:54] JM: So if I understand correctly, you pair IP addresses with identity on the private network.

[00:10:04] DC: That's right. Yes.

[00:10:06] JM: What are the complications or security risks to doing that?

[00:10:12] DC: Traditionally, IP-based security is a bad idea. This is an idea that went out a long time ago, and it went out because – I suppose for two reasons. The first is that it's very easy for intermediates on the network to generate packets as if they come from another IP address. Very easy depending on the network, sometimes it's harder. But in general, it's very hard to trust on a physical network that a person who's claiming to be a certain IP address is the person you thought they were a moment ago.

The second problem with IP-based security is multiuser systems. So if two people are logged into a UNIX server at the same time and one communicates with another server, like they write a shell script or they open a remote web browser or something like that, they will be using the IP address of that server to make to make the connection. And the other user on that machine will also be using the same IP address to make a connection. So these two problems make IP address security traditionally not very useful.

The reason why those two problems do not apply to Tailscale is, firstly, multiuser systems are much rarer than they used to be. There are far more computers than they were. And when we have multiuser systems, we tend to break them down into VMs or containers. And that means each user on the system tends to get their own network stack. It would definitely be insecure to use Tailscale traditional multiuser system and try and trust that every connection coming from

an IP address is one user if you let two people log in. We work hard to encourage people not to do that.

Installing Tailscale inside each VM on a multisystem is perfectly fine. Every user appears as a different IP address. The second reason is because we are pairing IP addresses with public-private key pairs, when we establish the underlying WireGuard tunnel between two machines on the network, the packets for a particular IP address have to come in through that tunnel using that cryptographic key pair.

And so the user has to be holding the private key that we associated with their identity in order to send packets as that IP address. And so that's how we work around these two traditional problems of IP-based security.

[00:12:44] JM: And just to bring people back to the motivation, what are the advantages of the IP-based security?

[00:12:55] DC: It's simple. Here's the short story. It's very easy to write a web server that uses the identity login system of one particular major company for like OAuth 2 if you use their existing libraries for doing it. And I say very easy. I mean, it's just a few hours' work. The problem is it takes me about five minutes to write a really trivial web server that I'd like to share with few people of a company. And so the very easy work of a few hours of authentication integration to make it safe to put on the internet so that a few people of the company can use it is a little bit ridiculous.

So there's this making wiring authentication into all of the applications we write. Inherently, it means we write lot less applications. It's just more of a pain to work with. So what we want to do is create a word where you have security by default where the only people who have access to this are people you've explicitly granted access to already.

My ideal personal project, small shell script equivalent server doesn't involve any security at all. Just anyone can access it can do what I need to do. And I only give access to people I trust. And that on the Internet today is a surprisingly difficult thing to do. And the goal is to make that easier.

[00:14:19] JM: Can you give some more specific pieces of why it's useful? So you have a diagram on your website that describes simplifying the connections between file servers, and Git servers, and Web servers, and local machines. Just explain those in more detail.

[00:14:42] DC: Right. So this comes down to a mesh architecture versus your more traditional VPN architecture. So traditionally, with the VPN, what you do is you install a VPN gateway, which is some server, which clients will then connect to. And you might install more than one gateway and you might configure subnets to route through multiple gateways. And this tends to involve some level of indirection.

Anecdotally, I use to use a corporate VPN at a major tech company to access internal resources. And one day the VPN went down, because the reasons things go down. I think a data set going down somewhere. And so I went into the VPN configuration and connected to the New York server. And the time I lived in Berkeley. And so I was connecting to the server in New York. Routing all my packets to it, which would then route back to the servers I was trying to reach in California. And that worked fine. And it works well enough that I forgot about it and forgot about it for a few months. And then they went and fixed the local VPN gateway, as they do, because it was a very, very competent company that had people who were on the ball. But of course, my network mostly works. So I just forgot about it. And I spent the next several months connecting to all of these local machines by routing through a machine in New York, which was adding over 100 milliseconds of latency to all of my typing as I working at a shell somewhere.

And so I've occasionally spent my time complaining about, "Oh, how slow our severs are. What a pain." And my colleagues would scratch their heads and say, "Oh, I don't know what's wrong with you. Seems fine to me." Until I eventually discovered this. I think I discovered this on a business trip to New York where the quality of my connection improved slightly to the machine back in California. I was like, "Why is the machine in California easier to reach from New York than it is when I'm in California? "And the answer is, "Well, I don't have to bounce all of my traffic to New York and back again. I just go straight from New York to there."

And this is the frustration of these manually configured networks, which involve indirecting

through other machines. One of the nice things about these automated mesh networks is you don't run into these misconfiguration issues. It finds the shortest path from where you are to the machine you're trying to access. So you don't have to struggle with these. And so that's why we have a diagram on our website showing all of these indirect lines certainly making direct connections to you.

[00:17:15] JM: How does the mesh network that you would build with Tailscale compare to a VPC? Like a virtual private cloud that you can build on AWS?

[00:17:30] DC: Oh well, that's a good question. It could come out looking exactly the same. If you created an AWS account and you spun up several AWS VM's and they roll on a VPC together and they were directly connected. And then you install Tailscale on each of those VM's, it would probably connect over the VPC. I suspect the mesh would find that as the shortest path and use it. And the packets would flow in very much the same way as they do through the VPC. And because the VPC effectively gives you complete control overall role of the IP addresses on there an all the machines that are assigned to it through the main AWS administration system, you get a lot of the same advantages you would get from Tailscale. You wouldn’t see a lot of difference there.

Where you would see differences are as soon as you try to go a bit beyond the VPC model, so you could, for example, then login to Tailscale on your phone and connect directly to any of those machines in the VPC that you’d install Tailscale on, and it would be like your phone is on the VPC immediately, which is quite nice. Similarly, a machine in your office or a machine in another cloud, different provider even, there might also be some small technical differences. I'm not extremely familiar with AWSs VPC implementation. I don't know if it encrypts traffic when you flow between availability zones. It could, but that would be a detail I have to look up. No Tailscale will encrypt all of the traffic between the machines on the VPC automatically. It doesn't know how to not encrypt traffic. But there are lots of similarities between the sorts of private networks you build with VPC and the sorts of private networks you build with Tailscale. The significant difference is Tailscale is software you can add ideally to just about any machine anywhere.

[00:19:24] JM: So let's say I am an admin for a technology company and I want to allow the

right users to access the right resources in my organization. I don't want to just give anyone access to anything across the entire VPN. How is access management done with Tailscale?

[00:19:48] DC: Yeah. This is interesting. So the user interface to this is, right now, a JSON file. We have an ACL file that people configure. And you can use it to define groups of users. You can tag machines with particular tags. And then you can describe the rules for who can access what. This group can access machines with this tag. Or you can write more specific rules about this IP address can reach the ports, these particular ports on this other IP address.

And this access control list, part of the control plane. So when every Tailscale client connects and identifies its public key and it's confirmed against an identity access management system, it gets down a list of all the public keys of its other nodes on its network it can talk to. It also gets a compiled version of this access control list. So we turn this high-level description of who can access what on a network into a set of firewall rules. This port on this IP can access this port on this IP. And send – And these are bespoke firewall rules for every single node on the network. So to make it more concrete, let's say we create a group called accounting and put three uses in it and then we tag a couple of service as the accounting servers and we write an ACL that says, “Only people in the accounting group can access the accounting servers.” Then when you login to Tailscale on your laptop, if your user is in the accounting group, then you will get down from Tailscale server the public keys and IP addresses of the accounting servers. And the accounting servers will get both your public key and a firewall rule saying that you are allowed to access resources on this accounting server. Users who are not in the accounting group will not have access to the accounting seven because of the rules we sent to the accounting server blocking it. So you can view it as a distributed firewall. Each machine is enforcing the rules about who is allowed to access it based on the global network rules.

[00:21:58] JM: What have been the biggest engineering problems you’ve had to solve in building Tailscale?

[00:22:03] DC: Oh! Well, there are lots of engineering problems. The biggest engineering problem of course is always hiring excellent engineers. It's never easy. Beyond that, there are several interesting technical things we did. We built Tailscale on WireGuard, which is in you VPN protocol. It’s a new VPN protocol is a fairway to describe it.

[00:22:25] JM: Is that open source?

[00:22:26] DC: Oh, yes. WireGuard is open source. It's created by a Linux kernel hacker named Jason Donenfield, and is a tunneling protocol I think would be probably the most accurate way to describe it. You can see all the details on wireguard.com. There's an implementation in the Linux kernel. Then I go, use a Lambda implementation, which is used in android, iOS apps, and a Mac OS client. And I believe there's now open BSD kernel implementation. And I assure other kernel implementations are following. It's really excellent protocol. It's wonderfully simple, very easy to think about. So we decided to build on that, even though that was relatively new at the time. The Linux kernel implementation hadn't gone upstream yet. And so we had to learn all that. And it's very interesting.

On top of that, we had sort of two technical difficulties we had to deal with. The first is the safe distribution of public keys for the nodes on your network. So this involved writing a server that could integrate with existing access management systems to demonstrate the identity of a user and then move their public key to other machines on the network.

The second big technical challenge was making the meshwork. And the hardest part of making a mesh work is getting the NAT traversal to work. So networks today have a feature that was implemented to deal with the fact that we were running out of IP addresses in the late 1990s, which has become sort of a security feature as well. So there's only 32 bits of IP addresses somewhere just north of 4 billion of them in IPV4, the primary specification leaves today for IP's. And this is just not enough for the number of machines that need IP addresses. And so the workaround is called our network address translation. And the way this works is that there are some small collections of private IPV4 addresses, which can be reused many, many times and never reach the public Internet. And whenever you set up a local network, the router on the local network has a DHCP server, which hands that these private IP address.

So right now, the machine I'm talking to you from has a private IP address. It's 192.168.1. something. I assume your machine also has some private IP address just like that. These private IP addresses never appear on the public Internet. Do you want to check and see what your IP address is? We could see if we have overlapping subnet ranges.

[00:24:54] JM: Sure. What do I do on Apple?

[00:24:55] DC: On Apple, you could open system preferences and then you go to the network icon. And then in network, there should be on network somewhere which has a green dot next to it, meaning it's on. If you click on it, it should print your IP address. Right. And I'm 192.168.166. Neither of these IP addresses work on the Internet. And yet somehow we're talking to each other through the Internet. This is where network address translation comes in.

So when packets from my machine reach the router in my house, it's takes the packet swaps off this 192.168 address and scribbles on the public IP address the ISP gave me and changes the port number to a port its assigned on the router. Sends it to you. Your router takes your public IP address. Scribbles it out and writes 192.168.4. whatever number you said. And then sends it on to your computer.

So there's this live translation of IP addresss going on. And this means the one IP address I have in my house can serve the dozen or so devices I have my house and the same on your end. This is how the IP address range got expanded for the whole network. But this is problematic for tunneling a software that wants to create a connection between my machine and yours.

So we're talking of a Zoom right now, and the way Zoom deals with this is easy. Our machines both connect with Zoom server on the Internet, and it has a public IP address. And so we reach out to it, and then all of our packets bounce through it. With Tailscale, it's more problematic, because if I had a client on my machine and you had a client on your machine, we are on the same Tailscale network. How do we get packets directly to each other? There'd be no address we could ride on the first packet and send to each other. So this requires a lot of NAT reversal techniques to make this work.

We have a long, very detailed blog post by my colleague, David Anderson, which goes into details on how this works on our website. The short story is we send packets to test servers on the Internet to try and work out what your public IP address is and what port it's mapped to. We then use a side channel to communicate what this address is to each other. So we have a

central server, which is our side channel for bouncing these addresses around. And then our machines attempt to talk directly to these temporary IP port pairs that your routers handed out.

This is technically pretty difficult. We're not the first to do it. WebRTC, the protocol, implements this. And we're following a playbook that's very similar to WebRTC, but it's relatively rare Internet technology in the sense that there aren't very many implementations of it around. And so that was technically very difficult on the creation of Tailscale. And it wasn't until my colleagues, David Anderson and Brad Fitzpatrick came on board that we really got that sorted out and working well. It works quite well now.

There are some other interesting technical asides. That NAT traversal technique that I described, or it's actually a family of techniques, works about 9 times out of 10. But there are always interesting networks. Interesting, if that's the word, where they won't work for very long tail of reasons. And on those networks, we use relay servers for moving packets around. So the encrypted packets from one Tailscale node to another can be bounced through one of our relay servers that are distributor around the world to reach the other side. That system had to be designed very carefully, because we don't want the ability to see anyone's traffic, because we don't want to be in that business. So we had to design it so that Tailscale clients could encrypt their own traffic in a way that we could not decrypt it. Send it to us, but we would know where it needs to go, and then pass it on. And we call that system dup, and it's responsible for making sure you can always create a connection if your Internet is working. Those of the major pieces of the puzzle that all have to come together to make Tailscale work well.

[00:28:58] JM: So it sounds like you don't want Tailscale to all be in the business of like if I'm a VPN operator knowing what is going on across my network.

[00:29:08] DC: We definitely don't want Tailscale to be able to see into your network. No. That doesn't seem – I don't think anyone wants that.

[00:29:15] JM: What about the admin? The admin of the network.

[00:29:19] DC: Oh sure. I mean, the admin is in control of the configuration of the network. They definitely could find a way to see into how a network is working. There are lots of things

they could – I mean, in our view, if you are setting up a network, it's your network and you set it up the way you want.

[00:29:36] JM: How do you audit the security of a VPN product?

[00:29:41] DC: Yeah. It's very hard. So at the heart of it, we divide the product in two, into control plane and the data. The data plane actually moves people's packets and is the final arbiter of whether key is the correct key to talk to and lean heavily on WireGuard to do this work for us. WireGuard is our data plane, and it's a really excellent piece of software that has been very carefully audited by many people, many security professionals. And that's really the only way to audit software correctly, is to make it open and to get lots of eyes on it, and to define it really carefully and make the definition small enough that a lot of people can load it into their heads and think hard about. And WireGuard has achieved that, and that's one of things I really like about it. It's so much simpler than all of the tunneling protocols that appeared before it, like IP sec, or TLS, or any of these other systems are much more complicated. It's WireGuard's simplicity that increases its security and a series of very good design choices along the way.

And so we rely heavily on that to do the heavy lifting of our auditing. Our control plane is a relatively new piece of software. We are very careful in how we design it so that we only move the least amount of information. We only move public keys. We never move private keys. We keep the protocol surface very small. We think very carefully about the design of any new features we're adding. And what we're working on right now is getting it ready for external audit by third parties.

If there's going to be a bug in the system, it's going to be in our control plane, because we're so confident in WireGuard as piece of software. And so that's where our work lies and not making sure it's secure.

[00:31:34] JM: What's your vision for the future of the product?

[00:31:37] DC: Our vision for the future of the product. Well, we want to go in several different directions. I couldn't tell you which direction is going to come first. The two big ones are we want to make it work for the long tail of complex interesting network designs out there. There are a lot

of features we need to add support complex networks. For example, one of my colleagues was just working on making sure the server can talk through web proxies on windows that require the [inaudible 00:32:01] tickets to be correct. There's a very long tail of features like that we need to get right.

The other direction we'd like to go is thinking about the product as a tool for developers writing software that I sort of mentioned earlier briefly. But to do this, we really need to think about interconnecting networks. Not just letting you build private networks, but how can you take a machine on your private network and share it with someone else? And there are several really exciting things we could do there that we just haven't done yet. And I think that is going to be a big focus of the more public-facing side of the product in the near future. How can I take Minecraft server running on my local network and share with my friends, to put it in fun terms.

[00:32:50] JM: Well, David, that sounds a great place to wrap up. Anything else you want to add?

[00:32:54] DC: Not for me. This was terrific. Thank you very much.

[END]