Episode Book

EPISODE #6350

Why Modern SSH Keys Cannot Fix Harvested Traffic

PUBLISHED	29 October 2026 at 18:19 UTC
DURATION	23:44
LISTEN	myweirdprompts.com/episode/ssh-keys-session-encryption-threats-55120/
HOSTS	Corn, Herman, Hilbert
SCRIPT MODEL	openai/gpt-6.1-sol
VOICES	chatterbox-regular (chatterbox-local)
PIPELINE	V5.25
SOURCES	13 cited by the research (listed at the end)
BOOK GENERATED	30 October 2026 at 07:19 UTC

SYNOPSIS

SSH user keys authenticate who you are, but they do not encrypt your session. That distinction matters against harvest-now, decrypt-later attacks, where stored traffic falls to quantum advances targeting ephemeral key exchange rather than your login credentials. A look at the differences between user keys, host authentication, and session negotiation explores what OpenSSH defaults actually protect, why hybrid signatures like ML-DSA require support at both endpoints, and why forward secrecy has its limits.

TRANSCRIPT



Corn

If somebody records your SSH connection today, is the key you pasted into GitHub the thing that could betray you later? Daniel wants to get underneath that familiar routine. Run ssh-keygen, accept the defaults, upload the public half, carry on. What have you actually chosen, and how much does that choice have to do with keeping the connection secret?



Herman

Less than people assume. The key you generate usually proves who you are. A separate exchange creates the secrets that encrypt your session. Those operations face different quantum threats, and replacing your login key doesn't automatically improve the other one.



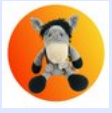
Corn

I like that the public key is the bit we all carefully inspect.



Corn

By which I mean we check that copying worked.



Herman

Usually the right level of attention for that particular operation. On current OpenSSH, running ssh-keygen without arguments generates Ed25519. Small, fast, well established. You haven't accidentally selected some antique because you pressed Enter.



Corn

There is comfort in discovering that my strongest cryptographic opinion was a carriage return.



Herman

The version matters, though. An older installed copy can have older defaults. We're talking about current OpenSSH, not every executable called ssh-keygen on every neglected laptop.



Corn

That immediately shifts what I'd check. I wouldn't begin by throwing away my key. I'd check what's actually installed at both ends. My laptop being current doesn't update the server, and a server being current doesn't upgrade whatever client an automation job has buried inside it.



Herman

You're already separating the endpoints, which is necessary. I'd separate the jobs too. Your user key authenticates you. The server has its own host key, which authenticates the server. Then ephemeral key exchange produces fresh session keys. The host signature binds that exchange to the server you're trying to reach.



Corn

Three things wearing the same word.



Herman

Yes, and they need different maintenance. You can have an excellent user key, an unverified host identity, and an old key exchange, all in one connection. Inspecting the public key file answers only one of those concerns.



Herman

Here's the development that changes the conversation. Post-quantum key exchange has been an OpenSSH default since version nine point zero, released in April twenty twenty-two. Version ten point six, released on October sixth, also enables a hybrid post-quantum signature option by default.



Corn

Enabled means available for negotiation, though. It doesn't mean every old server suddenly understands it, or that my existing Ed25519 key has acquired a new mathematical interior.



Herman

Your existing key stays exactly what it was. Support, selection, and possession are separate conditions. You need compatible implementations and the appropriate keys before the new signature algorithm can actually authenticate anything.



Corn

Herman, I can already see you preparing to replace every credential in the flat.



Herman

I'm preparing to explain the menu.



Corn

You explain menus with a shopping basket.



Herman

This time I'm keeping the old key too. You may consider that personal growth.



Herman

For Daniel's immediate choice, Ed25519 remains the straightforward new-key recommendation. Hardware-backed variants serve a different need. The new hybrid is useful where both endpoints support it. None of that replaces checking the negotiated connection, especially when the data needs to stay confidential for years.



Corn

Then the distinction worth keeping in view is what gets checked during login versus what gets used to encrypt the traffic. I can change one without changing the other, which is both useful and an excellent opportunity to fix the wrong thing.



Herman

The current key-generation menu contains RSA, ECDSA, Ed25519, the security-key versions of ECDSA and Ed25519, and the new ML-DSA forty-four combined with Ed25519. That's the complete current type list. DSA has disappeared from it.



Corn

Old tutorials remain remarkably committed to keeping DSA employed.



Herman

They're a reason to check the manual installed with your version. RSA remains supported and widely interoperable. Its security rests on the difficulty of factoring large integers. The current generation default is three thousand seventy-two bits, generally considered sufficient for classical security. The accepted minimum is one thousand twenty-four, which is a compatibility boundary, not my recommendation.



Corn

RSA being old doesn't make every RSA key broken. I'd be more interested in its size, where the private half has been copied, and whether I can revoke it without conducting an archaeological survey.



Herman

All relevant. ECDSA uses elliptic-curve discrete logarithms rather than integer factoring. It offers the NIST curves with sizes of two hundred fifty-six, three hundred eighty-four, or five hundred twenty-one bits. Those sizes don't compare directly with RSA bit lengths. Different mathematical problems need different key sizes.



Corn

Otherwise the RSA key wins by being longer on paper.



Herman

Ed25519 is elliptic-curve cryptography too, but with a fixed choice rather than a parameter menu. Compact keys, fast signatures, fewer decisions. It's current best practice for ordinary new OpenSSH user keys. ECDSA remains current, RSA remains useful for compatibility, and DSA belongs out of new deployments.



Herman

Corn, would you choose a hardware-backed key for your own login, knowing you'd have to keep the token nearby?



Corn

For a server I care about, probably. For an unattended job, no, unless somebody has solved the attendance part. My attraction is that stealing the laptop's files wouldn't also steal a reusable private key. My objection is that you would install it and then borrow the USB port for something musical.



Herman

I can use another port. The security-key variants, identified by the sk ending, use FIDO hardware. The signing secret stays on the token. Normally you touch it to authorize a signature. There's still a local key-handle file, but copying that file doesn't copy the token's private secret.



Corn

Physical presence becomes a dependency. That's valuable, provided the workflow actually has a person in it. A hardware key doesn't become unsuitable because it interrupts automation. It may be doing exactly the thing you bought it to do.



Herman

Touch requirements have configuration nuances, but yes, that's the intended distinction. Hardware backing protects against extracting the secret. It doesn't make the underlying signature mathematics quantum-resistant.



Corn

A locked drawer does not improve factoring.



Herman

The new hybrid addresses the mathematics. ML-DSA forty-four is a lattice-based signature standardized in FIPS two hundred four. OpenSSH combines it with Ed25519. Each signature contains both components, and both must verify.



Corn

I'd initially assumed the classical part was there so an older server could accept half.



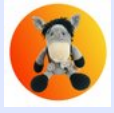
Herman

It can't. That's an important correction. The peer must understand the composite algorithm. The classical component is insurance against a weakness being discovered in the new mathematics. Breaking ML-DSA alone wouldn't let somebody forge the composite while Ed25519 remained secure.



Corn

You're enthusiastic about a new algorithm because they haven't trusted it alone.



Herman

Very enthusiastic. Cryptographers can be pleased and suspicious simultaneously.



Corn

You've finally found a hobby that accommodates both.



Herman

Let me put that key into an actual login. A typical client first offers the public key without a signature. The server checks whether that key is acceptable for the account, often through authorized_keys. If it is, the client makes a signed authentication request using the private half.



Corn

The offer saves me unlocking a key the server won't accept. That makes the passphrase prompt a local event, not a password I'm sending to the server.



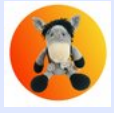
Herman

Right. Well... local unless you've arranged some external signing mechanism, but the passphrase still isn't an SSH account password sent for verification. An agent may already have the key unlocked, or a token may produce the signature. The signed request includes the session identifier, binding the proof to this connection.



Corn

Recording my successful login doesn't produce a reusable performance of me logging in.



Herman

The server checks authorization and verifies the signature over the session-bound data. Knowing your public key lets it verify that proof. It doesn't let the server, or an observer, sign as you.



Corn

Meanwhile, before it trusts my account claim, my client has to decide whether the server is actually the server. Otherwise I'm proving my identity very competently to the wrong machine.



Herman

The server sends its public host key and signs the exchange hash during key exchange. Your client checks that host identity against known_hosts. On first contact, the usual model is trust on first use, the familiar fingerprint prompt.



Corn

The prompt whose entire security department is me saying yes.



Herman

Unless you've obtained the expected fingerprint independently, yes. First contact establishes a remembered identity; later connections can detect a change. Corn, would you consider several new known_hosts entries after one acceptance suspicious?



Corn

I would have. If I agreed to one fingerprint and received several keys, I'd assume the interface had smuggled something past me.



Herman

OpenSSH can learn additional host keys after authentication through its host-key update extension. The server advertises them, and the client requests proof of possession of the corresponding private keys, bound to the established session. Subject to client settings, that can add several key types from one trusted host.



Corn

That gives rotation a way forward without asking me to approve an unexplained new identity every time. It also means counting lines in known_hosts isn't a reliable count of how many servers I've trusted.



Herman

Session encryption is separate again. Ephemeral key agreement establishes fresh shared secrets, from which the connection derives encryption and integrity keys. The temporary secrets are discarded. Later theft of the long-term authentication keys doesn't, by itself, decrypt captured past sessions. That's forward secrecy.



Corn

The default key choice is already modern, then. The unresolved part is what the connection actually negotiates.



Hilbert

Hilbert's Mid-Show Haiku.



Hilbert

For the intern in the corridor. A small key, chosen by no one. It opens the door anyway.



Corn

The corridor has permissive defaults.



Herman

Forward secrecy has a boundary, though. It protects against later theft of long-term secrets. It doesn't promise protection against a future mathematical attack on the recorded ephemeral exchange itself.



Corn

That's the gap I was missing. I hear that the temporary secrets were discarded and assume the evidence has gone. An observer still has the public exchange messages, and those could become enough if the underlying problem becomes solvable.



Herman

A sufficiently capable quantum computer could attack classical key agreement from those recorded messages. It could recover the session secrets and decrypt the stored traffic. That is harvest now, decrypt later. The attacker doesn't need to steal your login key along the way.



Corn

Which makes replacing my RSA login with Ed25519 a perfectly reasonable modernization that doesn't resolve this particular threat. Both are classical signatures anyway. Even replacing it with the new hybrid leaves the session exchange as something I need to inspect separately.



Herman

OpenSSH moved on that earlier. Its first default post-quantum exchange combined Streamlined NTRU Prime with X25519. Version ten point zero changed the default to the ML-KEM seven hundred sixty-eight hybrid with X25519, using SHA two hundred fifty-six.



Corn

The algorithm names sound like the password requirements escaped.



Herman

You'll see the full names in connection diagnostics. The useful thing to recognize is the hybrid construction. The new post-quantum mechanism and established classical agreement both contribute. A break in the new mechanism alone shouldn't leave you worse off than the classical baseline.



Corn

No need to bet every confidential session on the newest contestant.



Herman

Provided the implementation and combination are sound, yes. That's a narrower statement than saying hybrids cannot fail. They add implementation work and protocol complexity. They hedge uncertainty about the mathematical components.



Herman

Would you migrate login signatures at the same speed as key exchange, Corn?



Corn

I'd prioritize exchange for anything with a long secrecy lifetime. A database export copied today might still be sensitive when somebody can attack it. For signatures I'd want a planned retirement of classical credentials before forgery becomes practical, but a signature recorded today doesn't unlock today's encrypted export later.



Herman

That's the distinction OpenSSH makes explicitly. There's no analogous store-now-decrypt-later exposure from the signature itself. A future attacker able to forge your classical signature could impersonate you in a future login, or forge a host's identity for a new connection.



Corn

Still serious. Just a different clock. Authentication has to be ready before the attacker can forge. Confidentiality may need protection years before that, because the recording can happen now.



Herman

Exactly... no, more specifically, those timelines depend on how long your data matters and how long migration takes. Nobody has a dependable arrival date for a cryptographically relevant quantum computer. That means a machine able to break the deployed cryptography at useful scale, not merely one that demonstrates a quantum calculation.



Corn

A laboratory milestone doesn't automatically come with my server password.



Herman

OpenSSH describes estimates ranging from five to twenty years, with many expectations around the mid-twenty-thirties. That's a planning range, not a countdown. Waiting for certainty would give an attacker a larger archive of vulnerable traffic.



Corn

I'm comfortable adopting protection before we know whether it will become necessary. I'm less comfortable announcing that my whole estate is quantum-safe because one laptop got an update.



Herman

Negotiation makes that especially misleading. A capable client connecting to an old server can still fall back. Starting with OpenSSH ten point one, the client warns when the connection isn't using post-quantum key exchange and identifies the store-now-decrypt-later risk.



Corn

At least that warning names the operation. It gives me somewhere more useful to look than the age of my public key.



Herman

Ten point six also adds the weak-cryptography warning setting on the server side, allowing classical clients to be logged. If you deliberately keep a classical-only host, the client warning can be suppressed for that host. That changes notification, not protection.



Corn

The connection remains weak, but now quietly.



Herman

There's a more immediate way to lose access with ten point six. Its hybrid signature support drops the experimental vendor-suffixed algorithm name used in ten point four and ten point five. Those experimental keys need regeneration or removal.



Corn

This is where your shopping basket becomes relevant.



Herman

I said I'd keep the old key.



Corn

I heard you. I'm checking whether the server heard you.



Herman

Fair. The name appears in public-key entries, host-key records, and private-key files. Merely editing the visible public line isn't a complete migration. The new release doesn't recognize the old experimental name.



Corn

A failed login is inconvenient. A server configuration refusing to load is rather more committed.



Herman

An unknown name left in the accepted user-key or host-key algorithm settings can produce a fatal Bad key types error. The daemon may refuse to start. On a remote machine without console access, that's a lockout caused by the security upgrade.



Corn

Excellent. No quantum computer can log in now.



Herman

Before restarting, validate the server configuration with sshd's test mode. Inspect its effective host-key, accepted-user-key, and key-exchange settings. Keep a working session open, and have another route in. Retain an Ed25519 credential alongside the new hybrid for older servers and implementations that don't understand it.



Corn

Keeping a fallback sounds contradictory until you name what it protects against. It's protecting availability during migration. It also means I haven't retired classical authentication yet, so I shouldn't claim that I have.



Herman

Enabling support isn't retirement. You need an inventory of which credentials and clients remain, then a policy for removing them. That's difficult even in organizations with substantial budgets.



Corn

I'd expect the hidden clients to be worse than the visible servers. People remember the machine they log into. They forget the scheduled transfer that logs into it every night.



Herman

The federal picture suggests that inventory is a real bottleneck. An October GAO audit found only one of twenty-four agencies had completed its quantum-vulnerable cryptography inventory. Most also reported lacking cryptographic expertise.



Corn

Twenty-three agencies still looking for the keys.



Herman

New national security systems face quantum-resistant support requirements from twenty twenty-seven under CNSS Policy fifteen, with legacy phase-out by twenty thirty. Those deadlines make discovery operational work now. Before changing an algorithm, somebody has to find the client using it.



Corn

Especially when the client is an unattended laptop doing encryption with a key nobody remembers installing.



Hilbert

Authentication, Corn. That laptop key is doing authentication. You've spent most of this explaining the distinction, and then you gave it encryption at the end. I did the same thing this month. My old RSA key worked on a new ten point six server, and the server logged a warning about classical key exchange. I spent the afternoon blaming the key.



Herman

The RSA signature can succeed while an older client negotiates classical exchange. Those events fit together.



Hilbert

It was two thousand forty-eight bits. Passphrase on it. I'd copied it between laptops for more than a decade because it was already in all the authorized_keys files. I kept the copy on a USB stick marked Do not lose. The fire-safe copy said Do not lose, backup. This month I stopped carrying the stick. I erased it and gave it away.



Corn

The label made a fairly broad promise to its next owner.



Hilbert

A man I met travelling had shown me his security token. I still think about him because he carried it separately from his laptop. Mine had always been in the laptop bag. At home I use the separate key now. I don't like carrying the copy with the thing it's copied for. The warning was about exchange, as Herman said. I understood it after lunch.



Herman

Keeping the signing device separate also changes what happens if the entire bag disappears.



Hilbert

Unless you've put both in your coat.



Corn

Your stubborn old key had another advantage. You knew where it was authorized. A new credential makes you discover where the old one wasn't replaced.



Hilbert

I missed a Raspberry Pi when I added Ed25519 everywhere. Six months later I needed it. Password login was disabled. I took it out of the cupboard and mounted the card on another machine. That took longer than lunch.



Herman

That distinction between possessing a new key and deploying it everywhere matters for the hybrid migration too. The cryptography can be ready while a single forgotten endpoint still decides whether access works.



Corn

The missed machine also explains why removing the old credential should be its own deliberate step. Creating the replacement is easy to observe. Proving you've finished distributing it takes actual checking.



Herman

One adjacent detail we haven't used is private-key storage. The default OpenSSH format protects a passphrase-encrypted private key with bcrypt-based derivation, currently using thirty-two rounds. That's protection for the file at rest, separate again from signatures and session exchange.



Corn

The passphrase matters if somebody steals the file. It doesn't upgrade the network negotiation, and the network negotiation doesn't rescue an exposed signing key. Both can deserve attention without becoming the same job.



Herman

The migration I'd watch is the removal stage. Adding hybrid support is measurable; finding the last forgotten classical client before disabling it is harder. That cupboard machine is the sort of endpoint an inventory has to include.



Corn

Thanks to Hilbert Flumingtop for producing.



Herman

If you want more of this, try episode three sixty-eight, The Geometry of Secrets; episode thirty-five, The Privacy Gap; and episode three, Safetensors or something else. This has been My Weird Prompts. If you enjoyed this episode, leave us a review. It helps other listeners find the show.



Corn

See you tomorrow.

SOURCES

What the research for this episode read before the script was written. Primary sources first.

1. **OpenSSH post-quantum page [primary]**
   https://www.openssh.org/pq.html

2. **ssh-keygen(1), current OpenBSD man page [primary]**
   https://man.openbsd.org/ssh-keygen

3. **ssh_config(5) [primary]**
   https://man.openbsd.org/ssh_config

4. **OpenSSH release notes index, incl. 10.6 (2026-10-06) [primary]**
   https://www.openssh.org/releasenotes.html

5. **OpenSSH 10.6 release notes, 2026-10-06 [primary]**
   https://www.openssh.org/txt/release-10.6

6. **oss-security: Announce: OpenSSH 10.6 released, 2026-10-06 [primary]**
   https://www.openwall.com/lists/oss-security/2026/10/06/2

7. **IETF draft -00, 2026-06-02**
   https://datatracker.ietf.org/doc/html/draft-miller-sshm-mldsa44-ed25519-composite-sigs

8. **Help Net Security on OpenSSH 10.4, 2026-07-06**
   https://www.helpnetsecurity.com/2026/07/06/openssh-10-4-released/

9. **Help Net Security on OpenSSH 10.6, 2026-10-07**
   https://www.helpnetsecurity.com/2026/10/07/openssh-10-6-released/

10. **Enabling quantum-safe authentication in OpenSSH (walkthrough of host-key and user-key auth flows)**
    https://iyanmv.com/posts/openssh-pqc-auth/

11. **OpenSSH 10.6 turns on post-quantum signatures, 2026-10-12 (migration steps, NSA/GAO deadlines)**
    https://yaamlabs.com/blog/openssh-10-6-post-quantum-keys

12. **heise online on hybrid PQ signatures**
https://www.heise.de/en/news/OpenSSH-first-brings-hybrid-post-quantum-signatures-11356758.html

13. **LWN: OpenSSH 10.6 released**
https://lwn.net/Articles/1098980/