Who Owns RFC7217's Secret Key

Who Should Create Ipv6 Addresses?

The operating system (OS)? The Internet Service Provider (ISP)? The IT department? The end user? Hiveware makes a case for the latter. The IETF RFC7217 standard "A Method for Generating Semantically Opaque Interface Identifiers with Ipv6 Stateless Address Autoconfiguration (SLAAC)", contains a 128-bit Secret Key. The standard describes how to generate a secure IP address and it is this key that is its cardinal ingredient. Whoever creates this key, owns the data being transported by the IP address, and by extension, the message's content. With that in mind, who should own this key?

Not the OS

Fx, no file in the windows OS can be owned 100%. This is a fact stated in their documentation. An admin somewhere can always take control. This means that the OS can also take control, if directed to do so. Microsoft could fx surreptitiously put a back door in, fx, their CreateUnicastIpAddress function, and compromise it. If the reader thinks this is far-fetched, then think of these two examples. One: it is highly likely that in the US, the government will eventually pass a law that requires companies that produce encrypted messaging software to put in encryption backdoors for their use. Two: think of Huawei. The reason why the US doesn't want to do business with them is, the Chinese constitution states that all its companies on demand must turn over the ability to access customer's encrypted data or go out of

business. In light of this thinking, to think the OS is a benign entity is utterly naive. The Windows OS if it uses RFC7217 at all, merely throws the secret key away. It is certainly not available to the End User programmer.

Not the ISP

It is the FCC's responsibility to hold ISPs accountable for tranmitting ip packets fairly and with technical integrity around the globe. There are many issues, however. They could be commandeered by the government to compromise their security. Another is the standards committee could be co-oped which is the current case with the IETF 8200 standard where a sub-group is trying to compromise the standard by allowing big-interests to add data to the Ipv6 header information en route. See this appeal by other standards committee authors to halt this. Let's say their appeal fails, then that alone would be reason enough to want end-to-end security to be in the End User's hands.

Not the IT Department

""Why should an app create an ipv6 address, instead of letting the OS do it?" This represents the attitude of network IT personnel. Today OS networking calls can return a viable Ipv4 or Ipv6 address to the app. But is such an address secure? Hiveware doesn't think so.

IT Networking personnel are not programmers. They mostly do scripting. In a sense, they are middle-men because they are hired to configure and maintain networking infrastructure.

What are the reasons why the End User should own his data?

Reason 1: Secure Unit-value Transfer

If I want to be absolutely be sure that if I subtract fx $1 from an account at end point A and add $1 to an account at end point B, which is normal accounting, then there cannot be any chance now or in the future, of man-in-the-middle ownership.

Reason 2: Sales

The main reason is sales, however. In order to be able to tell prospective downstream customers that ownership of their data is by design, ownable. If the technology can't do that, it can't get them to believe that their data is safe from outside intervention any time in the future, as today's many data breaches bear witness to. Potential customers won't buy it. Case in point: decentralized applications, Dapps, today are all failing to one degree or another simply because they do not have a real distributed data model that is secure, and this is true because no one can really trust end point data. This is despite millions of dollars being spent by Venture and Investor Capitalists on these companies who are making those Dapps.

How does RFC7217 make sure the End User owns the new Ipv6 address?

RFC7217 lets the application be developed for the End User so that the IP addresses may be created programmatically by the owner and encrypted by the owner. The RFC7217 Secret Key never leaves the owner's control.

THE FUTURE OF IP NETWORKING AND ADDRESSES

A lot is going to happen with Ipv6 in the next 3 years when ISO C++23 Networking is supposed to come out. ISO may or may not make this happen by 2023. In the meantime, there is certainly a substantial market to be had in this area. The RFC7217Cpp library development has been beta complete and even stress-tested (100K round trips) with a full-blown app, Hiveware, on top. The library code was designed to be as close as possible to the ipv6 protocol that will be emerging by C++23. Ipv6 deployment is nascent. The RFC7217 library will evolve as the standards, deployment, ISPs, OSes and End User Apps evolve. Already, the RFC7217 library is designed to degrade gracefully starting with optimal to less-optimal IP interfaces.

CONCLUSION

The first OSI layer controllable by the programmer of Dapps is Layer 4 of the OSI model, the transport layer meaning TCP and eventually UDP. All low-level transport calls are ip family agnostic, that is, may be Ipv4 or Ipv6, and transport layer interface type agnostic, that is may be Ethernet or Wireless agnostic. But, the argument of this article is that it is explicitly Layer 1, the application layer, or End User who, by use of the RFC7217's Secret Key, is able to maintain item ownership in spite of all the threats outlined in the above.

RFC7217Cpp is a library that may be used in legacy, or new, centralized apps as well as distributed apps described here.

RFC7217Cpp lets you build (D)apps that don't have to be aware of what the quality of IP interface is and lets your company focus on the app part instead of the low-level data interchange part.

Hiveware® for RFC7217Cpp library is a part of the Hiveware engine, but is being made acquire-able as a separate library.

For more information contact us at rfc7217cpp@hiveware.com.