# Development of Remote Access Services —Evolution of ID Gateway

## 2.1 Introduction

At the end of September 2024, the ID Gateway Service IIJ had been providing for around 26 years was finally discontinued. Having begun in December 1998[*1], it was one of the longest-running services in IIJ's history.

The ID Gateway Service was a service that provided remote access, a technology for connecting to computers in remote locations and something that is essential for the now widespread practice of remote work. VPN technology is now considered essential for remote access, but the ID Gateway Service did not use VPN when it was first released in 1998. Even site-to-site VPNs were still only just starting to gain traction at the time, and while there were client-side VPN implementations, they could hardly have been called practical. Here, we look back on developments from this era through to the present, where VPN has become an essential technology in our society, alongside the history of IIJ's ID Gateway Service.

## 2.2 The Early Days of Remote Access Services

Around the time IIJ launched the ID Gateway Service in 1998, remote access meant installing network devices for dial-up access, such as the Ascend MAX, within an organization, dialing into it, and establishing an IP connection via PPP. The key with these dial-up routers was that they needed to provide internal connectivity, because if you simply wanted to connect to the Internet, you could connect to an ISP. At the time, however, such devices could not implement access controls such as firewalls, and there were apparently cases in which they connected to internal networks with virtually no restrictions.

IIJ, meanwhile, had been recommending to users that they keep their internal networks separate from the Internet, with firewalls at the boundary. Remote access that provided a back door around that separation was a vulnerability in and of itself, and IIJ believed that when providing remote access as a service, it should be incorporated within the same access control policy as firewalls. Moreover, instead of putting dial-up routers on the user's network, the system was designed to use IIJ's dial-up service. And thus ID Gateway 1.0 was developed to use existing Internet connectivity as the means for remote access while requiring firewall-like access control for access to internal networks.

## 2.3 ID Gateway 1.0

ID Gateway 1.0 was developed based on an application-level gateway firewall. The base OS was BSD/OS 3.1. With an application-level gateway firewall, a proxy intercepts all communications passing through the firewall and only relays subsequent communications if permitted according to access control rules. Since communications are relayed at the application level, protocols such as HTTP and SMTP are interpreted before the communications are forwarded. The access control portion of this application-level gateway firewall was extended to allow or deny communications for individual destinations based on the PPP account username (Figure 1).

We anticipated that it would often be necessary to grant permission for each specific destination rather than using a broad rule to allow access to the entire network simply because a user had been authenticated. This was because it had been pointed out through design reviews and internal beta testing that it would be crucial to protect not only connecting users but also the services within organizations to which communications were being directed. The team therefore aimed to implement fine-grained access control.

Let's examine the details a little more closely. As indicated in the lower left of Figure 2, when a user access request comes in, it is passed through the operating system to the relay program (proxy). At this point, the user's source IP address, destination IP address, and port number are passed to the relay program, which then queries the ID authentication daemon to determine whether the communication is permitted under the access control rules. Since

*1 "Launch of ID Gateway Service," iij.ad.jp, IIJ (October 2, 1998) (https://www.iij.ad.jp/news/pressrelease/1998/pdf/gateway.pdf, in Japanese).

access control rules are in Link-ID form (PPP account user ID), the ID authentication daemon needs to convert the source IP address to a Link-ID. If a previous query result exists in the cache, the ID authentication daemon immediately completes the conversion from source IP address to Link-ID using the cached information. If no such result exists in the cache, it sends a query to what is called IIJ's ID server. Communication with the ID authentication daemon and the ID server uses a protocol developed by IIJ called the Link-ID protocol.

Once the ID authentication daemon obtains the Link-ID, it determines whether the communication is permitted for the resolved user by assessing the access control rules and returns an allow or deny value to the relay program. This ensured that remote access was only allowed when permitted under the access control rules.

## 2.4 ID Gateway 2.0

With ID Gateway 2.0, released in September 2000, we overhauled the configuration user interface from a console application to a graphical user interface (GUI) called ID Gateway Policy Manager. After extensive discussion about how to make the access control rules more intuitive to configure, we eventually designed a unique spreadsheet-like interface for the access control rule panel in which users

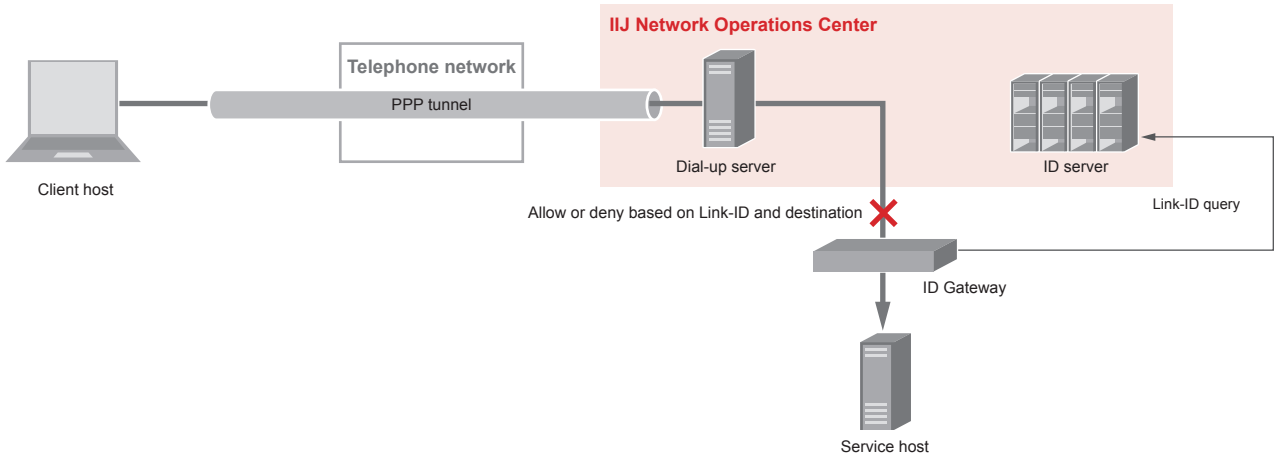


Figure 1: Overall Structure of ID Gateway 1.0Figure 1: Overall Structure of ID Gateway 1.0

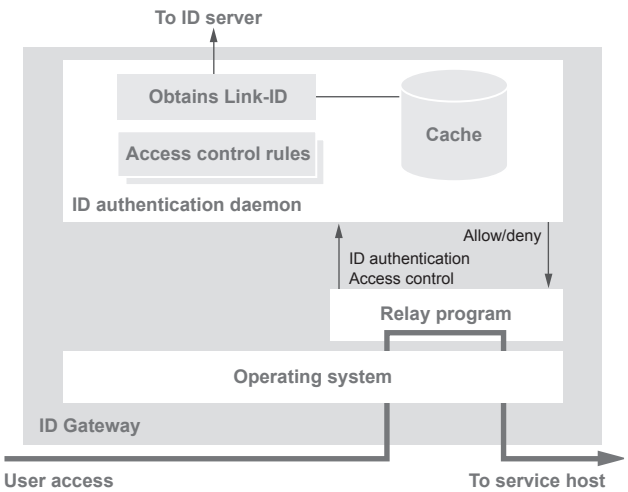


Figure 2: ID Gateway 1.0's Access Control

could mark cells with ○ or ×, as shown in Figure 3. We also switched to NetBSD 1.4 for the base OS.

## 2.5 ID Gateway 3.0

ID Gateway 3.0, released in April 2002, provided support for VPN connections based on PPPs such as L2TP/IPsec and PPTP. At the time, however, we had to overcome several challenges to make this happen.

We first needed to implement tunneling protocols L2TP and PPTP. Existing open-source implementations at the time used a process-forking model that used one process per tunnel, and at the anticipated scale of the ID Gateway Service, this would have involved hundreds of processes, which would have been impractical due to memory constraints. We had implemented the daemons on ID Gateway using an event-driven model since version 1.0, and the tunneling protocols also needed to be implemented with an event-driven model. We did this from scratch using C++.

Next, for the IPsec part of L2TP/IPsec, we were able to use the IKE and IPsec implementations from WIDE's KAME project*2, which had been incorporated into NetBSD. There was one problem, however. To use IPsec through NAT, you need to implement IPsec NAT-T, but the version incorporated into NetBSD 1.4 did not have this. Unless IPsec NAT-T was implemented, the UDP checksums would not match, and even if we were to bypass that, the problem was that only one host behind NAT would be able to connect. On the ID Gateway Service, we called this the "first come, only served" problem. For ID Gateway 3.0, we decided to accept this "first come, only served" problem as a given restriction and only solve the UDP checksum mismatch issue. Here, we decided to skip UDP checksum verification since ESP's HMAC is already used to perform a check on incoming transmissions. For outgoing transmissions, we set the optional UDP checksum field to 0, meaning it would not be used, so that checksum verification would also be skipped on the peer host.

Issues can potentially arise with PPTP when traversing NAT. PPTP uses TCP for control and GRE for transmitting data. There are no NAT traversal issues with the control portion since it uses TCP, but issues similar to those with L2TP/IPsec can arise with the GRE-based data transmissions. But implementations that used the Call-ID

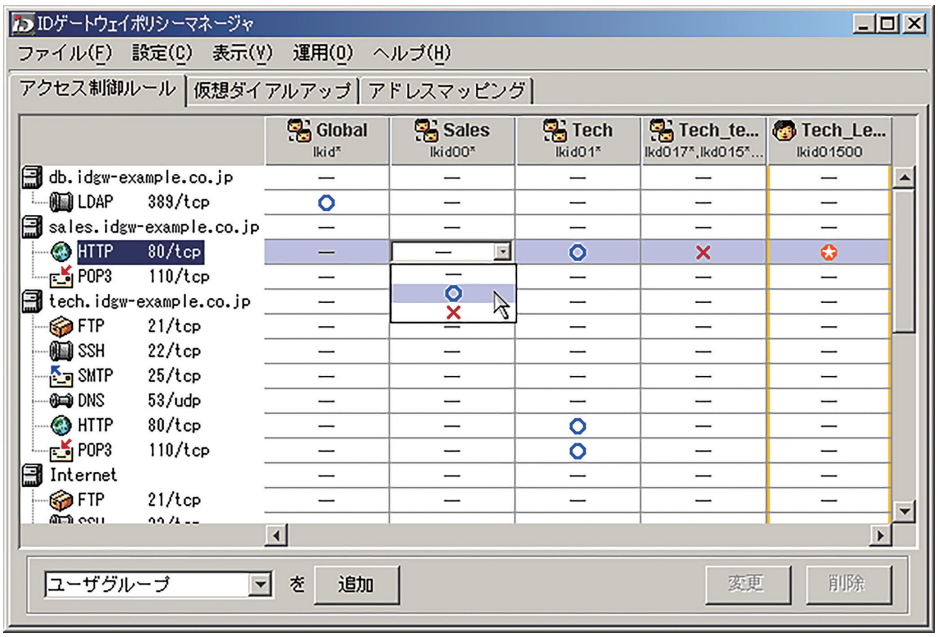


Figure 3: ID Gateway 2.0's Policy Manager

*2 KAME Project (https://www.kame.net/index.html).

in the GRE header for NAT masquerading had started to become widespread, particularly in consumer routers. We thus expected the "first come, only served" problem to be less of an issue with PPTP than with L2TP/IPsec. Additionally, PPTP effectively requires MPPE for packet encryption, and MPPE uses the proprietary encryption algorithm RC4. We therefore obtained a license to use RC4 from RSA Data Security.

For the PPP implementation, we decided to use FreeBSD's ppp, which was based on iij-ppp[*3] and had been extended with Multi-PPP to enable multiple PPP connections over multiple lines, allowing a single process to act as the endpoint for multiple PPP connections, plus it had originally been developed by IIJ.

We also had to figure out how to handle VPN authentication. For a typical VPN service, you would first connect to the user directory service for authentication. But with ID Gateway up to version 2.0, users had been using dial-up connections, so existing users were already on IIJ's dial-up service, and we had issued PPP accounts to connecting users with access control rules being configured for those accounts. We realized that using the same accounts for VPN authentication would make it possible to use various existing components as well, so that is how we implemented it. Authentication requests from the PPP daemon on the ID Gateway were proxied by the ID authentication daemon and authenticated by the RADIUS server on IIJ's ID server. This allowed users to configure a single PPP account in the same way both for dial-up and for VPN entries via L2TP/IPsec and PPTP. As Figure 4 shows, we were also able to implement the access control mechanism in almost the same manner as before. Because we designed the system so that VPN would be handled in the same way as conventional dial-up, within the ID Gateway Service we called this virtual dial-up (VDIP) and referred to conventional dial-up as real dial-up.

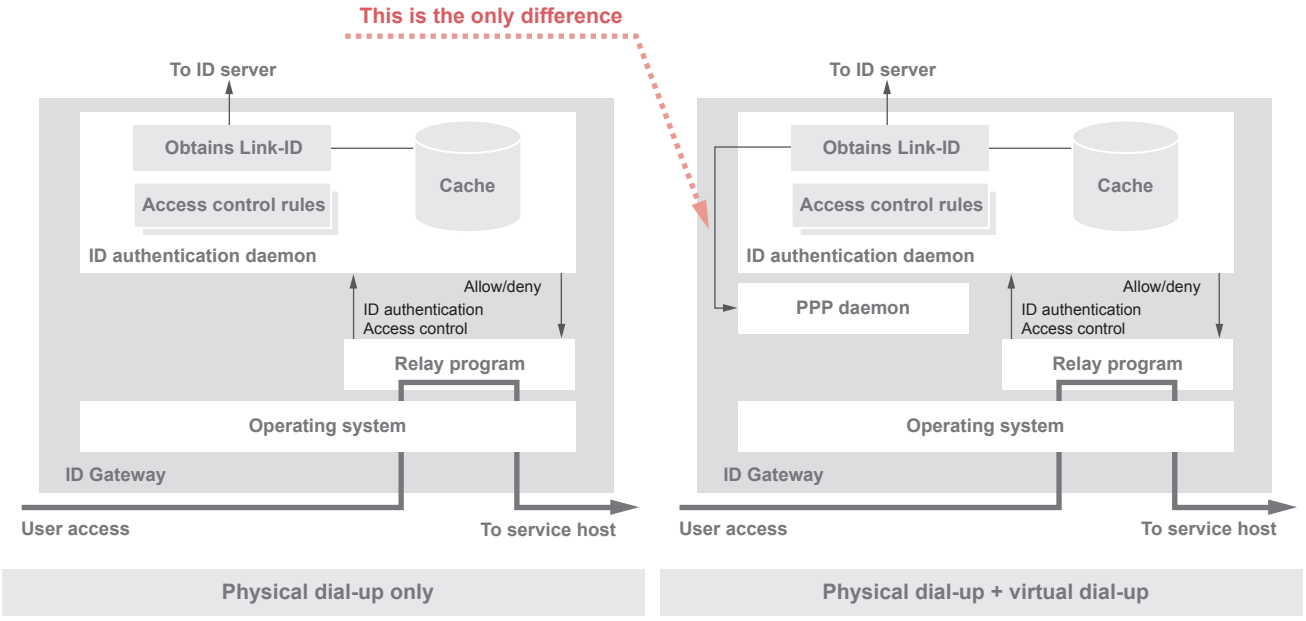


Figure 4: Dial-up Method and Access Control

*3 An open-source PPP implementation developed by Toshiharu Ohno and others at IIJ. It was widely used from the late 1990s to the early 2000s, particularly among BSD users.

## 2.6 ID Gateway 4

With ID Gateway 4.00, released in July 2005, we added an authentication server integration feature to allow the use of a RADIUS or LDAP server from the user's network—which could be an Active Directory environment, for example—as the virtual dial-up authentication server. We also overhauled the reporting system and added a rule viewer function.

In ID Gateway 4.02, released in April 2006, we implemented a new VPN protocol called SSL Dial-up (SSLDIP). As noted earlier, L2TP/IPsec and PPTP can experience problems when used across NAT. And these protocols are sometimes completely unusable when only a limited range of ports (e.g., HTTP, HTTPS, DNS) can be used, such as from within restricted organizational networks or networks available at overseas hotels. Meanwhile, SSL-VPN products were beginning to appear on the market, and since SSL-VPN products using SSL for transport are entirely unaffected by these problems, we knew we had to implement equivalent functionality in ID Gateway.

We designed SSLDIP to use SSL (now TLS) as the transport layer and create an L2 tunnel between the client and ID Gateway, with PPTP running on top of that. The aim was to use PPP as the common base for the tunnels that were ultimately created, allowing us to otherwise use the same mechanisms as before, including for authentication and access control. For the L2 tunnel, we decided to use the open-source OpenVPN, and we developed a Windows client to make configuring the system and managing connections easy.

For ID Gateway 4.02, we also rewrote the implementations of the L2TP/IPsec and PPTP server functions from scratch and created a new daemon. In the original implementation, the VPN tunnel processing code was written in C++ rather than C, which caused problems when porting to embedded environments like the SEIL series[*4] (IIJ's series of proprietary high-performance routers for enterprises). The PPP portion had also previously been a separate program, and we simplified this by rewriting it with the bare minimum functionality and incorporated it into the same program as the VPN, improving performance and maintainability. This program was npppd[*5], which would later be incorporated into OpenBSD. Further, the RADIUS portion repurposed code originally used in the ID server, and this served as the prototype for what is now the OpenBSD RADIUS library[*6].

## 2.7 ID Gateway 5

In ID Gateway 5.00, released in March 2008, we added a client authentication feature. This provided additional authentication on top of the initial VPN connection authentication, making it equivalent to what is now called multi-factor authentication. This feature addressed user requests for the ability to restrict the range of devices able to connect and for contingencies to mitigate the impact of password leaks. We extended the ID authentication daemon and relay program to implement the terminal authentication feature. After a VPN connection was established and up until the point that terminal authentication was completed, the ID Gateway terminal authentication feature restricted access to DNS and authentication pages only, and once terminal authentication was successful, it would switch to the full set of access control rules. This method of

*4 SEIL (https://www.seil.jp/, in Japanese).

*5 src/usr.sbin/npppd/, github.com, GitHub (https://github.com/openbsd/src/tree/8b2d863473/usr.sbin/npppd/).

*6 src/lib/libradius/, github.com, GitHub (https://github.com/openbsd/src/tree/8b2d863473/lib/libradius/).

access restriction is equivalent to what is now called a captive portal. The authentication mechanism worked by redirecting web access from the client to an authentication page. An applet on the authentication page would send the client device's MAC address to the ID Gateway, which would then check it against the list of registered MAC addresses stored in the ID Gateway's database to verify that the connection was coming from a legitimate device owned by an authorized VPN user.

In addition, to enable hot standby, we added VRRP functionality by porting it from SEIL. We also added support for EAP authentication in the authentication server integration feature.

With ID Gateway 5.02, released in December 2009, we brought the operating system source code and the SEIL/X series source code together to unify the codebase. This enabled IPsec NAT-T and thus resolved the long-standing issue of not being able to accommodate multiple L2TP/IPsec users behind NAT. We switched to NetBSD 3.1 for the base OS.

## 2.8 ID Gateway 6

ID Gateway 6.00, released in November 2011, added support for SSTP as a new tunneling protocol. SSTP is a protocol developed by Microsoft that operates over TLS and tunnels PPP frames in a manner similar to L2TP/IPsec and the like. As it is TLS-based, there are no issues connecting from behind NAT, and since the client is included as standard in Windows, we did not need to distribute our own client. And because it is PPP-based, we were able to reuse the existing authentication and access control components as is. Although SSTP is an open protocol, we needed to obtain a license from Microsoft for commercial use.

## 2.9 The End of ID Gateway Development and Challenges Faced

The initial technical challenges in terms of connecting from behind NAT that we faced when we launched the virtual dial-up service were resolved with the addition of support for IPsec NAT-T in version 5.02 and SSTP in version 6.00. Load per user was continuing to increase year by year, however, meaning we could no longer achieve the desired performance on a single gateway.

The first conceivable factor here is that ID Gateway was, from the outset, application gateway software, not a router. All communications were relayed at the application layer, and it basically did not perform IP forwarding. This inevitably results in higher loads compared with IP-level forwarding. What we should have had was a mechanism for offloading to IP-level forwarding on a case-by-case basis.

Another factor was that the relay program providing the application gateway functionality was designed to run as a single process, so it was unable to make use of multiple CPUs even when they were present, and it had issues with multi-core support.

The final factor to consider is the base OS kernel. The base OS of the final version of ID Gateway is NetBSD 3.1, but work on multi-core support for its network stack had not yet started, and thus similar to the relay program, it

was unable to use multiple CPUs even when present, and it had issues with multi-core support.

These seemingly separate issues are in fact nothing more than the manifestation of software obsolescence. Modern software continues to evolve across the globe via the Internet, and systems are bound to become outdated if neglected. Looking back, we now realize that the issues with the ID Gateway software stem from us not having taken steps to address this inevitable software obsolescence.

We determined that extending ID Gateway to resolve these issues would be difficult for various reasons, and so we decided to discontinue development of the ID Gateway software and pass the torch to its successor service, IIJ GIO Remote Access Service, along with Tornado, a new gateway OS developed at IIJ.

## 2.10 IIJ GIO Remote Access Service and Tornado

In February 2013, we launched a new remote access service called IIJ GIO Remote Access Service (GAM). GAM is a cloud-based service with the VPN gateway running in IIJ's cloud. For the VPN gateway, we use Tornado, a system newly developed at IIJ to replace ID Gateway.

Tornado is IIJ's in-house gateway software integrating network-related software functions within IIJ, developed based on OpenBSD. We needed the base OS for the successor to ID Gateway to provide enterprise firewall-level packet filtering capabilities and the ability to implement kernel extensions for transparent proxies, and OpenBSD met all these requirements at the time development began in 2011. Moreover, from the outset it also provided features we had wanted for ID Gateway, such as socket splicing, which moves the work of relaying packets from the application level back into the kernel, and VRF for policy routing and the like. It also incorporated the npppd daemon from ID Gateway.

In Tornado, we replaced the relay program with a new multi-core compatible daemon. And in the access control rules, we made it possible to use a single configuration item to switch between application-level relaying and IP-level forwarding through packet filtering.

A major difference between ID Gateway and Tornado is that while ID Gateway was software exclusively for the ID Gateway service, Tornado is general-purpose software. It is designed so that service-specific functions that cannot be standardized are developed as optional packages.

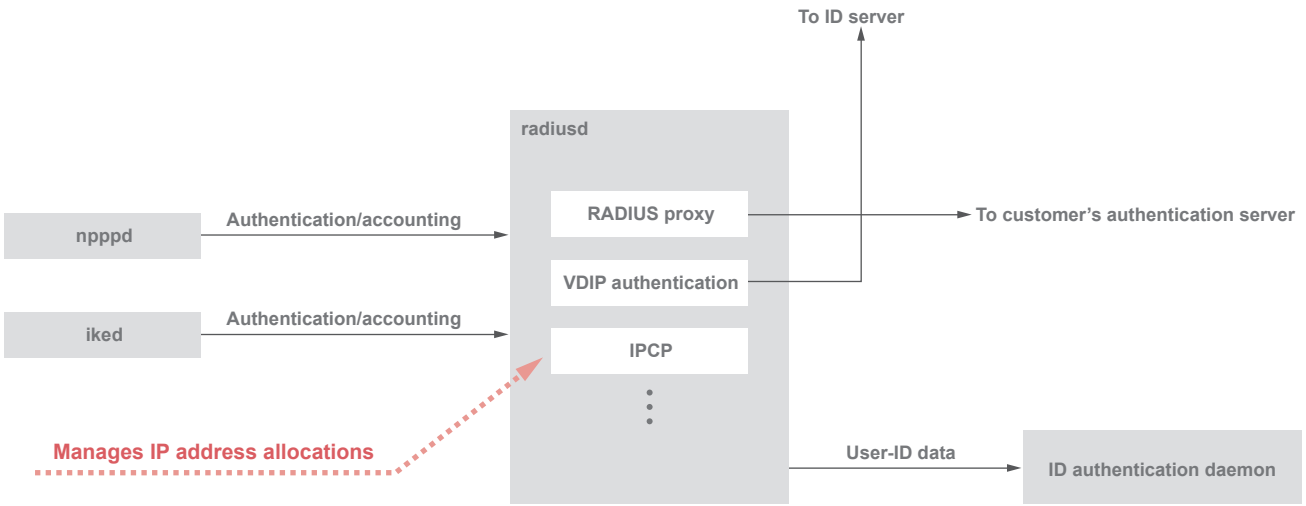


Figure 5: Overview of Connections Using IKEv2

Service-specific features are subject to a relatively high amount of churn, with new features constantly being desired while old ones become obsolete. We designed the system so that such features could be added and removed easily.

In December 2024, GAM added support for connections using the new VPN protocol IKEv2. As IKEv2 is not a PPP-based protocol, the authentication and access control mechanisms used by other protocols could not be used with the IKEv2 daemon available in OpenBSD at that time. Creating a system entirely separate from the PPP-based one to solve this problem would have meant two different methods had to be managed and maintained. With Tornado, we decided to standardize by adding RADIUS authentication and accounting capabilities to the IKEv2 daemon. We went with a unified authentication system in RADIUS and, as shown in Figure 5, implemented the system such that a local RADIUS daemon centrally manages the allocation of IP addresses to VPNs, and the access control mechanism thus works in the same way for both PPP-based VPNs and IKEv2.

Internally, we use Tornado version 4.5, which is based on an OpenBSD version released about a year ago, so we are using a relatively recent version. Having learnt from the ID Gateway experience, we now update the base OS version regularly as part of continuous integration, so we no longer end up being stuck with stale versions of the base OS.

## 2.11 Conclusion

Looking back, ID Gateway was never just a remote access service. It provided security features whereby users were authenticated via PPP accounts and only able to engage in the communications permitted under the access control rules for their authenticated IDs. With ID Gateway 5, we also added device-level authentication. These features constitute what is now called a Zero Trust Architecture (ZTA).

The history of ID Gateway and the subsequent transition to IIJ GIO Remote Access Service is also a story of in-house software development at IIJ. Tornado is IIJ's current infrastructure software and successor to ID Gateway. Looking forward, we will continue to work with new technologies and strive to provide even better services that address the changing needs of users as well as changes in the broader landscape.

**Masahiko Yasuoka**
System Development Section 1, Applied Technology Development Department, System Development Division, Network Services Business Unit, IIJ
Mr. Yasuoka joined IIJ in 1998. After developing ID Gateway and other systems, he proposed and developed Tornado, a gateway OS integrating the functionality of IIJ's internal software. He continues to develop and maintain this system today.