Network Working Group
Request for Comments: 5505
Category: Informational

B. Aboba
D. Thaler
L. Andersson
S. Cheshire
Internet Architecture Board

May 2009

Principles of Internet Host Configuration

Status of This Memo

This memo provides information for the Internet community. It does not specify an Internet standard of any kind. Distribution of this memo is unlimited.

Copyright Notice

Abstract

This document describes principles of Internet host configuration. It covers issues relating to configuration of Internet-layer parameters, as well as parameters affecting higher-layer protocols.

Table of Contents

1. Introduction

This document describes principles of Internet host [STD3] configuration. It covers issues relating to configuration of Internet-layer parameters, as well as parameters affecting higher-layer protocols.

In recent years, a number of architectural questions have arisen, for which we provide guidance to protocol developers:

- The protocol layers and general approaches that are most appropriate for configuration of various parameters.

- The relationship between parameter configuration and service discovery.

- The relationship between per-interface and per-host configuration.

- The relationship between network access authentication and host configuration.

- The desirability of supporting self-configuration of parameters or avoiding parameter configuration altogether.

- The role of link-layer protocols and tunneling protocols in Internet host configuration.

The role of the link-layer and tunneling protocols is particularly important, since it can affect the properties of a link as seen by higher layers (for example, whether privacy extensions [RFC4941] are available to applications).

1.1. Terminology

link

A communication facility or medium over which nodes can communicate at the link layer, i.e., the layer immediately below IP. Examples are Ethernets (simple or bridged), Point-to-Point Protocol (PPP) links, X.25, Frame Relay, or ATM networks as well as Internet- or higher-layer "tunnels", such as tunnels over IPv4 or IPv6 itself.

on link

An address that is assigned to an interface on a specified link.

off link

The opposite of "on link"; an address that is not assigned to any interfaces on the specified link.

mobility agent

Either a home agent or a foreign agent [RFC3344] [RFC3775].

1.2. Internet Host Configuration

1.2.1. Internet-Layer Configuration

Internet-layer configuration is defined as the configuration required to support the operation of the Internet layer. This includes configuration of per-interface and per-host parameters, including IP address(es), subnet prefix(es), default gateway(s), mobility agent(s), boot service configuration and other parameters:

IP address(es)

Internet Protocol (IP) address configuration includes both configuration of link-scope addresses as well as global addresses. Configuration of IP addresses is a vital step, since practically all of IP networking relies on the assumption that hosts have IP address(es) associated with (each of) their active network interface(s). Used as the source address of an IP packet, these IP addresses indicate the sender of the packet; used as the destination address of a unicast IP packet, these IP addresses indicate the intended receiver.

The only common example of IP-based protocols operating without an IP address involves address configuration, such as the use of DHCPv4 [RFC2131] to obtain an address. In this case, by definition, DHCPv4 is operating before the host has an IPv4 address, so the DHCP protocol designers had the choice of either using IP without an IP address, or not using IP at all. The benefits of making IPv4 self-reliant, configuring itself using its own IPv4 packets, instead of depending on some other protocol, outweighed the drawbacks of having to use IP in this constrained mode. Use of IP for purposes other than address configuration can safely assume that the host will have one or more IP addresses, which may be self-configured link-local addresses [RFC3927] [RFC4862], or other addresses configured via DHCP or other means.

Subnet prefix(es)

Once a subnet prefix is configured on an interface, hosts with an IP address can exchange unicast IP packets directly with on-link hosts within the same subnet prefix.

Default gateway(s)

Once a default gateway is configured on an interface, hosts with an IP address can send unicast IP packets to that gateway for forwarding to off-link hosts.

Mobility agent(s)

While Mobile IPv4 [RFC3344] and Mobile IPv6 [RFC3775] include their own mechanisms for locating home agents, it is also possible for mobile nodes to utilize dynamic home agent configuration.

Boot service configuration

Boot service configuration is defined as the configuration necessary for a host to obtain and perhaps also to verify an appropriate boot image. This is appropriate for disk-less hosts looking to obtain a boot image via mechanisms such as the Trivial File Transfer Protocol (TFTP) [RFC1350], Network File System (NFS) [RFC3530], and Internet Small Computer Systems Interface (iSCSI) [RFC3720] [RFC4173]. It also may be useful in situations where it is necessary to update the boot image of a host that supports a disk, such as in the Preboot Execution Environment [PXE] [RFC4578]. While strictly speaking, boot services operate above the Internet layer, where boot service is used to obtain the Internet-layer code, it may be considered part of Internet-layer configuration. While boot service parameters may be provided on a per-interface basis, loading and verification of a boot image affects behavior of the host as a whole.

Other IP parameters

Internet-layer parameter configuration also includes configuration of per-host parameters (e.g., hostname) and per-interface parameters (e.g., IP Time-To-Live (TTL) to use in outgoing packets, enabling/disabling of IP forwarding and source routing, and Maximum Transmission Unit (MTU)).

1.2.2. Higher-Layer Configuration

Higher-layer configuration is defined as the configuration required to support the operation of other components above the Internet-layer. This includes, for example:

Name Service Configuration

The configuration required for the host to resolve names. This includes configuration of the addresses of name resolution servers, including IEN 116 [IEN116], Domain Name System (DNS), Windows Internet Name Service (WINS), Internet Storage Name Service (iSNS) [RFC4171] [RFC4174], and Network Information Service (NIS) servers [RFC3898], and the setting of name resolution parameters such as the DNS domain and search list [RFC3397], the NetBIOS node type, etc. It may also include the transmission or setting of the host’s own name. Note that link-local name resolution services (such as NetBIOS [RFC1001], Link-Local Multicast Name Resolution (LLMNR) [RFC4795], and multicast DNS (mDNS) [mDNS]) typically do not require configuration.

Once the host has completed name service configuration, it is capable of resolving names using name resolution protocols that require configuration. This not only allows the host to communicate with off-link hosts whose IP addresses are not known, but, to the extent that name services requiring configuration are utilized for service discovery, also enables the host to discover services available on the network or elsewhere. While name service parameters can be provided on a per-interface basis, their configuration will typically affect behavior of the host as a whole.

Time Service Configuration

Time service configuration includes configuration of servers for protocols such as the Simple Network Time Protocol (SNTP) and the Network Time Protocol (NTP). Since accurate determination of the time may be important to operation of the applications running on the host (including security services), configuration of time servers may be a prerequisite for higher-layer operation. However, it is typically not a requirement for Internet-layer configuration. While time service parameters can be provided on a per-interface basis, their configuration will typically affect behavior of the host as a whole.

Other service configuration

This can include discovery of additional servers and devices, such as printers, Session Initiation Protocol (SIP) proxies, etc. This configuration will typically apply to the entire host.

2. Principles

This section describes basic principles of Internet host configuration.

2.1. Minimize Configuration

Anything that can be configured can be misconfigured. Section 3.8 of "Architectural Principles of the Internet" [RFC1958] states: "Avoid options and parameters whenever possible. Any options and parameters should be configured or negotiated dynamically rather than manually."

That is, to minimize the possibility of configuration errors, parameters should be automatically computed (or at least have reasonable defaults) whenever possible. For example, the Path Maximum Transmission Unit (PMTU) can be discovered, as described in "Packetization Layer Path MTU Discovery" [RFC4821], "TCP Problems with Path MTU Discovery" [RFC2923], "Path MTU discovery" [RFC1191], and "Path MTU Discovery for IP version 6" [RFC1981].

Having a protocol design with many configurable parameters increases the possibilities for misconfiguration of those parameters, resulting in failures or other sub-optimal operation. Eliminating or reducing configurable parameters helps lessen this risk. Where configurable parameters are necessary or desirable, protocols can reduce the risk of human error by making these parameters self-configuring, such as by using capability negotiation within the protocol, or by automated discovery of other hosts that implement the same protocol.

2.2. Less Is More

The availability of standardized, simple mechanisms for general-purpose Internet host configuration is highly desirable. "Architectural Principles of the Internet" [RFC1958] states, "Performance and cost must be considered as well as functionality" and "Keep it simple. When in doubt during design, choose the simplest solution."

To allow protocol support in many types of devices, it is important to minimize the footprint requirement. For example, IP-based protocols are used on a wide range of devices, from supercomputers to small low-cost devices running "embedded" operating systems. Since the resources (e.g., memory and code size) available for host configuration may be very small, it is desirable for a host to be able to configure itself in as simple a manner as possible.

One interesting example is IP support in preboot execution environments. Since by definition boot configuration is required in hosts that have not yet fully booted, it is often necessary for pre-boot code to be executed from Read Only Memory (ROM), with minimal available memory. Many hosts do not have enough space in this ROM for even a simple implementation of TCP, so in the Preboot Execution Environment (PXE) the task of obtaining a boot image is performed using the User Datagram Protocol over IP (UDP/IP) [RFC768] instead. This is one reason why Internet-layer configuration mechanisms typically depend only on IP and UDP. After obtaining the boot image, the host will have the full facilities of TCP/IP available to it, including support for reliable transport protocols, IPsec, etc.

In order to reduce complexity, it is desirable for Internet-layer configuration mechanisms to avoid dependencies on higher layers. Since embedded devices may be severely constrained on how much code they can fit within their ROM, designing a configuration mechanism in such a way that it requires the availability of higher-layer facilities may make that configuration mechanism unusable in such devices. In fact, it cannot even be guaranteed that all Internet-layer facilities will be available. For example, the minimal version of IP in a host’s boot ROM may not implement IP fragmentation and reassembly.

2.3. Minimize Diversity

The number of host configuration mechanisms should be minimized. Diversity in Internet host configuration mechanisms presents several problems:

Interoperability

As configuration diversity increases, it becomes likely that a host will not support the configuration mechanism(s) available on the network to which it has attached, creating interoperability problems.

Footprint

For maximum interoperability, a host would need to implement all configuration mechanisms used on all the link layers it supports. This increases the required footprint, a burden for embedded devices. It also leads to lower quality, since testing resources (both formal testing, and real-world operational use) are spread more thinly -- the more different configuration mechanisms a device supports, the less testing each one is likely to undergo.

Redundancy

To support diversity in host configuration mechanisms, operators would need to support multiple configuration services to ensure that hosts connecting to their networks could configure themselves. This represents an additional expense for little benefit.

Latency

As configuration diversity increases, hosts supporting multiple configuration mechanisms may spend increasing effort to determine which mechanism(s) are supported. This adds to configuration latency.

Conflicts

Whenever multiple mechanisms are available, it is possible that multiple configurations will be returned. To handle this, hosts would need to merge potentially conflicting configurations. This would require conflict-resolution logic, such as ranking of potential configuration sources, increasing implementation complexity.

Additional traffic

To limit configuration latency, hosts may simultaneously attempt to obtain configuration by multiple mechanisms. This can result in increasing on-the-wire traffic, both from use of multiple mechanisms as well as from retransmissions within configuration mechanisms not implemented on the network.

Security

Support for multiple configuration mechanisms increases the attack surface without any benefit.

2.4. Lower-Layer Independence

"Architectural Principles of the Internet" [RFC1958] states, "Modularity is good. If you can keep things separate, do so."

It is becoming increasingly common for hosts to support multiple network access mechanisms, including dialup, wireless, and wired local area networks; wireless metropolitan and wide area networks; etc. The proliferation of network access mechanisms makes it desirable for hosts to be able to configure themselves on multiple networks without adding configuration code specific to each new link layer.

As a result, it is highly desirable for Internet host configuration mechanisms to be independent of the underlying lower layer. That is, only the link-layer protocol (whether it be a physical link or a virtual tunnel link) should be explicitly aware of link-layer parameters (although those link-layer parameters may be configured by general Internet-layer mechanisms). Introduction of lower-layer dependencies increases the likelihood of interoperability problems and adds Internet-layer configuration mechanisms that hosts need to implement.

Lower-layer dependencies can be best avoided by keeping Internet host configuration above the link layer, thereby enabling configuration to be handled for any link layer that supports IP. In order to provide media independence, Internet host configuration mechanisms should be link-layer protocol independent.

While there are examples of Internet-layer configuration within the link layer (such as in PPP IPv4CP [RFC1332] and "Mobile radio interface Layer 3 specification; Core network protocols; Stage 3 (Release 5)" [3GPP-24.008]), this approach has disadvantages. These include the extra complexity of implementing different mechanisms on different link layers and the difficulty in adding new higher-layer parameters that would require defining a mechanism in each link-layer protocol.

For example, "PPP Internet Protocol Control Protocol Extensions for Name Server Addresses" [RFC1877] was developed prior to the definition of the DHCPINFORM message in "Dynamic Host Configuration Protocol" [RFC2131]; at that time, Dynamic Host Configuration Protocol (DHCP) servers had not been widely implemented on access devices or deployed in service provider networks. While the design of IPv4CP was appropriate in 1992, it should not be taken as an example that new link-layer technologies should emulate. Indeed, in order to "actively advance PPP’s most useful extensions to full standard, while defending against further enhancements of questionable value", "IANA Considerations for the Point-to-Point Protocol (PPP)" [RFC3818] changed the allocation of PPP numbers (including IPv4CP extensions) so as to no longer be "first come first served".

In IPv6, where link-layer-independent mechanisms such as stateless autoconfiguration [RFC4862] and stateless DHCPv6 [RFC3736] are available, PPP IPv6CP [RFC5072] configures an Interface-Identifier that is similar to a Media Access Control (MAC) address. This enables PPP IPv6CP to avoid duplicating DHCPv6 functionality.

However, Internet Key Exchange Version 2 (IKEv2) [RFC4306] utilizes the same approach as PPP IPv4CP by defining a Configuration Payload for Internet host configuration for both IPv4 and IPv6. While the IKEv2 approach reduces the number of packet exchanges, "Dynamic Host Configuration Protocol (DHCPv4) Configuration of IPsec Tunnel Mode" [RFC3456] points out that leveraging DHCP has advantages in terms of address management integration, address pool management, reconfiguration, and fail-over.

Extensions to link-layer protocols for the purpose of Internet-, transport-, or application-layer configuration (including server configuration) should be avoided. Such extensions can negatively affect the properties of a link as seen by higher layers. For example, if a link-layer protocol (or tunneling protocol) configures individual IPv6 addresses and precludes using any other addresses, then applications that want to use privacy extensions [RFC4941] may not function well. Similar issues may arise for other types of addresses, such as Cryptographically Generated Addresses [RFC3972].

Avoiding lower-layer dependencies is desirable even where the lower layer is link independent. For example, while the Extensible Authentication Protocol (EAP) may be run over any link satisfying its requirements (see Section 3.1 of [RFC3748]), many link layers do not support EAP and therefore Internet-layer configuration mechanisms that depend on EAP would not be usable on links that support IP but not EAP.

2.5. Configuration Is Not Access Control

Network access authentication and authorization is a distinct problem from Internet host configuration. Therefore, network access authentication and authorization is best handled independently of the Internet and higher-layer configuration mechanisms.

Having an Internet- or higher-layer protocol authenticate clients is appropriate to prevent resource exhaustion of a scarce resource on the server (such as IP addresses or prefixes), but not for preventing hosts from obtaining access to a link. If the user can manually configure the host, requiring authentication in order to obtain configuration parameters (such as an IP address) has little value. Network administrators who wish to control access to a link can better achieve this using technologies like Port-Based Network Access

Control [IEEE-802.1X]. Note that client authentication is not required for Stateless DHCPv6 [RFC3736] since it does not result in allocation of any limited resources on the server.

3. Additional Discussion

3.1. Reliance on General-Purpose Mechanisms

Protocols should either be self-configuring (especially where fate sharing is important), or use general-purpose configuration mechanisms (such as DHCP or a service discovery protocol, as noted in Section 3.2). The choice should be made taking into account the architectural principles discussed in Section 2.

Taking into account the general-purpose configuration mechanisms currently available, we see little need for development of additional general-purpose configuration mechanisms.

When defining a new host parameter, protocol designers should first consider whether configuration is indeed necessary (see Section 2.1).

If configuration is necessary, in addition to considering fate sharing (see Section 3.2.1), protocol designers should consider:

1. The organizational implications for administrators. For example, routers and servers are often administered by different sets of individuals, so that configuring a router with server parameters may require cross-group collaboration.

2. Whether the need is to configure a set of interchangeable servers or to select a particular server satisfying a set of criteria. See Section 3.2.

3. Whether IP address(es) should be configured, or name(s). See Section 3.3.

4. If IP address(es) are configured, whether IPv4 and IPv6 addresses should be configured simultaneously or separately. See Section 3.4.

5. Whether the parameter is a per-interface or a per-host parameter. For example, configuration protocols such as DHCP run on a per-interface basis and hence are more appropriate for per-interface parameters.

6. How per-interface configuration affects host-wide behavior. For example, whether the host should select a subset of the per-interface configurations, or whether the configurations are to merged, and if so, how this is done. See Section 3.5.

3.2. Relationship between IP Configuration and Service Discovery

Higher-layer configuration often includes configuring server addresses. The question arises as to how this differs from "service discovery" as provided by Service Discovery protocols such as "Service Location Protocol, Version 2" (SLPv2) [RFC2608] or "DNS-Based Service Discovery" (DNS-SD) [DNS-SD].

In Internet host configuration mechanisms such as DHCP, if multiple server instances are provided, they are considered interchangeable. For example, in a list of time servers, the servers are considered interchangeable because they all provide the exact same service -- telling you the current time. In a list of local caching DNS servers, the servers are considered interchangeable because they all should give you the same answer to any DNS query. In service discovery protocols, on the other hand, a host desires to find a server satisfying a particular set of criteria, which may vary by request. When printing a document, it is not the case that any printer will do. The speed, capabilities, and physical location of the printer matter to the user.

Information learned via DHCP is typically learned once, at boot time, and after that may be updated only infrequently (e.g., on DHCP lease renewal), if at all. This makes DHCP appropriate for information that is relatively static and unchanging over these time intervals. Boot-time discovery of server addresses is appropriate for service types where there are a small number of interchangeable servers that are of interest to a large number of clients. For example, listing time servers in a DHCP packet is appropriate because an organization may typically have only two or three time servers, and most hosts will be able to make use of that service. Listing all the printers or file servers at an organization is a lot less useful, because the list may contain hundreds or thousands of entries, and on a given day a given user may not use any of the printers in that list.

Service discovery protocols can support discovery of servers on the Internet, not just those within the local administrative domain. For example, see "Remote Service Discovery in the Service Location Protocol (SLP) via DNS SRV" [RFC3832] and DNS-Based Service Discovery [DNS-SD]. Internet host configuration mechanisms such as DHCP, on the other hand, typically assume the server or servers in the local administrative domain contain the authoritative set of information.

For the service discovery problem (i.e., where the criteria varies on a per-request basis, even from the same host), protocols should either be self-discovering (if fate sharing is critical), or use a general-purpose service discovery mechanism.

In order to avoid a dependency on multicast routing, it is necessary for a host to either restrict discovery to services on the local link or to discover the location of a Directory Agent (DA). Since the DA may not be available on the local link, service discovery beyond the local link is typically dependent on a mechanism for configuring the DA address or name. As a result, service discovery protocols can typically not be relied upon for obtaining basic Internet-layer configuration, although they can be used to obtain higher-layer configuration parameters.

3.2.1. Fate Sharing

If a server (or set of servers) is needed to get a set of configuration parameters, "fate sharing" (Section 2.3 of [RFC1958]) is preserved if those parameters are ones that cannot be usefully used without those servers being available. In this case, successfully obtaining those parameters via other means has little benefit if they cannot be used because the required servers are not available. The possibility of incorrect information being configured is minimized if there is only one machine that is authoritative for the information (i.e., there is no need to keep multiple authoritative servers in sync). For example, learning default gateways via Router Advertisements provides perfect fate sharing. That is, gateway addresses can be obtained if and only if they can actually be used. Similarly, obtaining DNS server configuration from a DNS server would provide fate sharing since the configuration would only be obtainable if the DNS server were available.

While fate sharing is a desirable property of a configuration mechanism, in some situations fate sharing may not be possible. When utilized to discover services on the local link, service discovery protocols typically provide for fate sharing, since hosts providing service information typically also provide the services. However, this is no longer the case when service discovery is assisted by a Directory Agent (DA). First of all, the DA’s list of operational servers may not be current, so it is possible that the DA may provide clients with service information that is out of date. For example, a DA’s response to a client’s service discovery query may contain stale information about servers that are no longer operational. Similarly, recently introduced servers might not yet have registered themselves with the DA. Furthermore, the use of a DA for service discovery also introduces a dependency on whether the DA is operational, even though the DA is typically not involved in the delivery of the service.

Similar limitations exist for other server-based configuration mechanisms such as DHCP. Typically DHCP servers do not check for the liveness of the configuration information they provide, and do not discover new configuration information automatically. As a result, there is no guarantee that configuration information will be current.

Section 3.3 of "IPv6 Host Configuration of DNS Server Information Approaches" [RFC4339] discusses the use of well-known anycast addresses for discovery of DNS servers. The use of anycast addresses enables fate sharing, even where the anycast address is provided by an unrelated server. However, in order to be universally useful, this approach would require allocation of one or more well-known anycast addresses for each service. Configuration of more than one anycast address is desirable to allow the client to fail over faster than would be possible from routing protocol convergence.

3.3. Discovering Names vs. Addresses

In discovering servers other than name resolution servers, it is possible to either discover the IP addresses of the server(s), or to discover names, each of which may resolve to a list of addresses.

It is typically more efficient to obtain the list of addresses directly, since this avoids the extra name resolution steps and accompanying latency. On the other hand, where servers are mobile, the name-to-address binding may change, requiring a fresh set of addresses to be obtained. Where the configuration mechanism does not support fate sharing (e.g., DHCP), providing a name rather than an address can simplify operations, assuming that the server’s new address is manually or automatically updated in the DNS; in this case, there is no need to re-do parameter configuration, since the name is still valid. Where fate sharing is supported (e.g., service discovery protocols), a fresh address can be obtained by re-initiating parameter configuration.

In providing the IP addresses for a set of servers, it is desirable to distinguish which IP addresses belong to which servers. If a server IP address is unreachable, this enables the host to try the IP address of another server, rather than another IP address of the same server, in case the server is down. This can be enabled by distinguishing which addresses belong to the same server.

3.4. Dual-Stack Issues

One use for learning a list of interchangeable server addresses is for fault tolerance, in case one or more of the servers are unresponsive. Hosts will typically try the addresses in turn, only attempting to use the second and subsequent addresses in the list if the first one fails to respond quickly enough. In such cases, having the list sorted in order of expected likelihood of success will help clients get results faster. For hosts that support both IPv4 and IPv6, it is desirable to obtain both IPv4 and IPv6 server addresses within a single list. Obtaining IPv4 and IPv6 addresses in separate lists, without indicating which server(s) they correspond to, requires the host to use a heuristic to merge the lists.

For example, assume there are two servers, A and B, each with one IPv4 address and one IPv6 address. If the first address the host should try is (say) the IPv6 address of server A, then the second address the host should try, if the first one fails, would generally be the IPv4 address of server B. This is because the failure of the first address could be due to either server A being down, or some problem with the host’s IPv6 address, or a problem with connectivity to server A. Trying the IPv4 address next is preferred since the reachability of the IPv4 address is independent of all potential failure causes.

If the list of IPv4 server addresses were obtained separately from the list of IPv6 server addresses, a host trying to merge the lists would not know which IPv4 addresses belonged to the same server as the IPv6 address it just tried. This can be solved either by explicitly distinguishing which addresses belong to which server or, more simply, by configuring the host with a combined list of both IPv4 and IPv6 addresses. Note that the same issue can arise with any mechanism (e.g., DHCP, DNS, etc.) for obtaining server IP addresses.

Configuring a combined list of both IPv4 and IPv6 addresses gives the configuration mechanism control over the ordering of addresses, as compared with configuring a name and allowing the host resolver to determine the address list ordering. See "Dynamic Host Configuration Protocol (DHCP): IPv4 and IPv6 Dual-Stack Issues" [RFC4477] for more discussion of dual-stack issues in the context of DHCP.

3.5. Relationship between Per-Interface and Per-Host Configuration

Parameters that are configured or acquired on a per-interface basis can affect behavior of the host as a whole. Where only a single configuration can be applied to a host, the host may need to prioritize the per-interface configuration information in some way (e.g., most trusted to least trusted). If the host needs to merge per-interface configuration to produce a host-wide configuration, it may need to take the union of the per-host configuration parameters and order them in some way (e.g., highest speed interface to lowest speed interface). Which procedure is to be applied and how this is accomplished may vary depending on the parameter being configured. Examples include:

Boot service configuration

While boot service configuration can be provided on multiple interfaces, a given host may be limited in the number of boot loads that it can handle simultaneously. For example, a host not supporting virtualization may only be capable of handling a single boot load at a time, or a host capable of supporting N virtual machines may only be capable of handling up to N simultaneous boot loads. As a result, a host may need to select which boot load(s) it will act on, out of those configured on a per-interface basis. This requires that the host prioritize them (e.g., most to least trusted).

Name service configuration

While name service configuration is provided on a per-interface basis, name resolution configuration typically will affect behavior of the host as a whole. For example, given the configuration of DNS server addresses and searchlist parameters on each interface, the host determines what sequence of name service queries is to be sent on which interfaces.

Since the algorithms used to determine per-host behavior based on per-interface configuration can affect interoperability, it is important for these algorithms to be understood by implementers. We therefore recommend that documents defining per-interface mechanisms for acquiring per-host configuration (e.g., DHCP or IPv6 Router Advertisement options) include guidance on how to deal with multiple interfaces. This may include discussions of the following items:

1. Merging. How are per-interface configurations combined to produce a per-host configuration? Is a single configuration selected, or is the union of the configurations taken?

2. Prioritization. Are the per-interface configurations prioritized as part of the merge process? If so, what are some of the considerations to be taken into account in prioritization?

4. Security Considerations

Secure IP configuration presents a number of challenges. In addition to denial-of-service and man-in-the-middle attacks, attacks on configuration mechanisms may target particular parameters. For example, attackers may target DNS server configuration in order to support subsequent phishing or pharming attacks such as those described in "New trojan in mass DNS hijack" [DNSTrojan]. A number of issues exist with various classes of parameters, as discussed in Section 2.6, Section 4.2.7 of "IPv6 Neighbor Discovery (ND) Trust

Models and Threats" [RFC3756], Section 1.1 of "Authentication for DHCP Messages" [RFC3118], and Section 23 of "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)" [RFC3315]. Given the potential vulnerabilities, hosts often restrict support for DHCP options to the minimum set required to provide basic TCP/IP configuration.

Since boot configuration determines the boot image to be run by the host, a successful attack on boot configuration could result in an attacker gaining complete control over a host. As a result, it is particularly important that boot configuration be secured. Approaches to boot configuration security are described in "Bootstrapping Clients using the Internet Small Computer System Interface (iSCSI) Protocol" [RFC4173] and "Preboot Execution Environment (PXE) Specification" [PXE].

4.1. Configuration Authentication

The techniques available for securing Internet-layer configuration are limited. While it is technically possible to perform a very limited subset of IP networking operations without an IP address, the capabilities are severely restricted. A host without an IP address cannot receive conventional unicast IP packets, only IP packets sent to the broadcast or a multicast address. Configuration of an IP address enables the use of IP fragmentation; packets sent from the unknown address cannot be reliably reassembled, since fragments from multiple hosts using the unknown address might be reassembled into a single IP packet. Without an IP address, it is not possible to take advantage of security facilities such as IPsec, specified in "Security Architecture for the Internet Protocol" [RFC4301] or Transport Layer Security (TLS) [RFC5246]. As a result, configuration security is typically implemented within the configuration protocols themselves.

PPP [RFC1661] does not support secure negotiation within IPv4CP [RFC1332] or IPv6CP [RFC5072], enabling an attacker with access to the link to subvert the negotiation. In contrast, IKEv2 [RFC4306] provides encryption, integrity, and replay protection for configuration exchanges.

Where configuration packets are only expected to originate on particular links or from particular hosts, filtering can help control configuration spoofing. For example, a wireless access point usually has no reason to forward broadcast DHCP DISCOVER packets to its wireless clients, and usually should drop any DHCP OFFER packets received from those wireless clients, since, generally speaking, wireless clients should be requesting addresses from the network, not offering them. To prevent spoofing, communication between the DHCP relay and servers can be authenticated and integrity protected using a mechanism such as IPsec.

Internet-layer secure configuration mechanisms include SEcure Neighbor Discovery (SEND) [RFC3971] for IPv6 stateless address autoconfiguration [RFC4862], or DHCP authentication for stateful address configuration. DHCPv4 [RFC2131] initially did not include support for security; this was added in "Authentication for DHCP Messages" [RFC3118]. DHCPv6 [RFC3315] included security support. However, DHCP authentication is not widely implemented for either DHCPv4 or DHCPv6.

Higher-layer configuration can make use of a wider range of security techniques. When DHCP authentication is supported, higher-layer configuration parameters provided by DHCP can be secured. However, even if a host does not support DHCPv6 authentication, higher-layer configuration via Stateless DHCPv6 [RFC3736] can still be secured with IPsec.

Possible exceptions can exist where security facilities are not available until later in the boot process. It may be difficult to secure boot configuration even once the Internet layer has been configured, if security functionality is not available until after boot configuration has been completed. For example, it is possible that Kerberos, IPsec, or TLS will not be available until later in the boot process; see "Bootstrapping Clients using the Internet Small Computer System Interface (iSCSI) Protocol" [RFC4173] for discussion.

Where public key cryptography is used to authenticate and integrity-protect configuration, hosts need to be configured with trust anchors in order to validate received configuration messages. For a node that visits multiple administrative domains, acquiring the required trust anchors may be difficult.

5. Informative References

[3GPP-24.008] 3GPP TS 24.008 V5.8.0, "Mobile radio interface Layer 3 specification; Core network protocols; Stage 3 (Release 5)", June 2003.

[DNSTrojan] Goodin, D., "New trojan in mass DNS hijack", The Register, December 5, 2008, http://www.theregister.co.uk/2008/12/05/new_dnschanger_hijacks/

[IEN116] J. Postel, "Internet Name Server", IEN 116, August 1979, http://www.ietf.org/rfc/ien/ien116.txt

[IEEE-802.1X] Institute of Electrical and Electronics Engineers, "Local and Metropolitan Area Networks: Port-Based Network Access Control", IEEE Standard 802.1X-2004, December 2004.

[DNS-SD] Cheshire, S., and M. Krochmal, "DNS-Based Service Discovery", Work in Progress, September 2008.

[mDNS] Cheshire, S. and M. Krochmal, "Multicast DNS", Work in Progress, September 2008.

[PXE] Henry, M. and M. Johnston, "Preboot Execution Environment (PXE) Specification", September 1999, http://www.pix.net/software/pxeboot/archive/pxespec.pdf

[RFC768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, August 1980.

[RFC1001] NetBIOS Working Group in the Defense Advanced Research Projects Agency, Internet Activities Board, and End-to-End Services Task Force, "Protocol standard for a NetBIOS service on a TCP/UDP transport: Concepts and methods", STD 19, RFC 1001, March 1987.

[RFC1191] Mogul, J. and S. Deering, "Path MTU discovery", RFC 1191, November 1990.

[RFC1332] McGregor, G., "The PPP Internet Protocol Control Protocol (IPCP)", RFC 1332, May 1992.

[RFC1350] Sollins, K., "The TFTP Protocol (Revision 2)", STD 33, RFC 1350, July 1992.

[RFC1661] Simpson, W., Ed., "The Point-to-Point Protocol (PPP)", STD 51, RFC 1661, July 1994.

[RFC1877] Cobb, S., "PPP Internet Protocol Control Protocol Extensions for Name Server Addresses", RFC 1877, December 1995.

[RFC1958] Carpenter, B., Ed., "Architectural Principles of the Internet", RFC 1958, June 1996.

[RFC1981] McCann, J., Deering, S., and J. Mogul, "Path MTU Discovery for IP version 6", RFC 1981, August 1996.

[RFC2131] Droms, R., "Dynamic Host Configuration Protocol", RFC 2131, March 1997.

[RFC2608] Guttman, E., Perkins, C., Veizades, J., and M. Day, "Service Location Protocol, Version 2", RFC 2608, June 1999.

[RFC2923] Lahey, K., "TCP Problems with Path MTU Discovery", RFC 2923, September 2000.

[RFC3118] Droms, R., Ed., and W. Arbaugh, Ed., "Authentication for DHCP Messages", RFC 3118, June 2001.

[RFC3315] Droms, R., Ed., Bound, J., Volz, B., Lemon, T., Perkins, C., and M. Carney, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 3315, July 2003.

[RFC3344] Perkins, C., Ed., "IP Mobility Support for IPv4", RFC 3344, August 2002.

[RFC3397] Aboba, B. and S. Cheshire, "Dynamic Host Configuration Protocol (DHCP) Domain Search Option", RFC 3397, November 2002.

[RFC3456] Patel, B., Aboba, B., Kelly, S., and V. Gupta, "Dynamic Host Configuration Protocol (DHCPv4) Configuration of IPsec Tunnel Mode", RFC 3456, January 2003.

[RFC3530] Shepler, S., Callaghan, B., Robinson, D., Thurlow, R., Beame, C., Eisler, M., and D. Noveck, "Network File System (NFS) version 4 Protocol", RFC 3530, April 2003.

[RFC3720] Satran, J., Meth, K., Sapuntzakis, C., Chadalapaka, M., and E. Zeidner, "Internet Small Computer Systems Interface (iSCSI)", RFC 3720, April 2004.

[RFC3736] Droms, R., "Stateless Dynamic Host Configuration Protocol (DHCP) Service for IPv6", RFC 3736, April 2004.

[RFC3748] Aboba, B., Blunk, L., Vollbrecht, J., Carlson, J., and H. Levkowetz, Ed., "Extensible Authentication Protocol (EAP)", RFC 3748, June 2004.

[RFC3756] Nikander, P., Ed., Kempf, J., and E. Nordmark, "IPv6 Neighbor Discovery (ND) Trust Models and Threats", RFC 3756, May 2004.

[RFC3775] Johnson, D., Perkins, C., and J. Arkko, "Mobility Support in IPv6", RFC 3775, June 2004.

[RFC3818] Schryver, V., "IANA Considerations for the Point-to-Point Protocol (PPP)", BCP 88, RFC 3818, June 2004.

[RFC3832] Zhao, W., Schulzrinne, H., Guttman, E., Bisdikian, C., and W. Jerome, "Remote Service Discovery in the Service Location Protocol (SLP) via DNS SRV", RFC 3832, July 2004.

[RFC3898] Kalusivalingam, V., "Network Information Service (NIS) Configuration Options for Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 3898, October 2004.

[RFC3927] Cheshire, S., Aboba, B., and E. Guttman, "Dynamic Configuration of IPv4 Link-Local Addresses", RFC 3927, May 2005.

[RFC3971] Arkko, J., Ed., Kempf, J., Zill, B., and P. Nikander, "SEcure Neighbor Discovery (SEND)", RFC 3971, March 2005.

[RFC3972] Aura, T., "Cryptographically Generated Addresses (CGA)", RFC 3972, March 2005.

[RFC4171] Tseng, J., Gibbons, K., Travostino, F., Du Laney, C., and J. Souza, "Internet Storage Name Service (iSNS)", RFC 4171, September 2005.

[RFC4173] Sarkar, P., Missimer, D., and C. Sapuntzakis, "Bootstrapping Clients using the Internet Small Computer System Interface (iSCSI) Protocol", RFC 4173, September 2005.

[RFC4174] Monia, C., Tseng, J., and K. Gibbons, "The IPv4 Dynamic Host Configuration Protocol (DHCP) Option for the Internet Storage Name Service", RFC 4174, September 2005.

[RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, December 2005.

[RFC4306] Kaufman, C., Ed., "Internet Key Exchange (IKEv2) Protocol", RFC 4306, December 2005.

[RFC4339] Jeong, J., Ed., "IPv6 Host Configuration of DNS Server Information Approaches", RFC 4339, February 2006.

[RFC4477] Chown, T., Venaas, S., and C. Strauf, "Dynamic Host Configuration Protocol (DHCP): IPv4 and IPv6 Dual-Stack Issues", RFC 4477, May 2006.

[RFC4578] Johnston, M. and S. Venaas, Ed., "Dynamic Host Configuration Protocol (DHCP) Options for the Intel Preboot eXecution Environment (PXE)", RFC 4578, November 2006.

[RFC4795] Aboba, B., Thaler, D., and L. Esibov, "Link-local Multicast Name Resolution (LLMNR)", RFC 4795, January 2007.

[RFC4821] Mathis, M. and J. Heffner, "Packetization Layer Path MTU Discovery", RFC 4821, March 2007.

[RFC4862] Thomson, S., Narten, T., and T. Jinmei, "IPv6 Stateless Address Autoconfiguration", RFC 4862, September 2007.

[RFC4941] Narten, T., Draves, R., and S. Krishnan, "Privacy Extensions for Stateless Address Autoconfiguration in IPv6", RFC 4941, September 2007.

[RFC5072] Varada, S., Ed., Haskins, D., and E. Allen, "IP Version 6 over PPP", RFC 5072, September 2007.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

[STD3] Braden, R., Ed., "Requirements for Internet Hosts - Communication Layers", STD 3, RFC 1122, October 1989.

Braden, R., Ed., "Requirements for Internet Hosts - Application and Support", STD 3, RFC 1123, October 1989.

Appendix A. Acknowledgments

Elwyn Davies, Bob Hinden, Pasi Eronen, Jari Arkko, Pekka Savola, James Kempf, Ted Hardie, and Alfred Hoenes provided valuable input on this document.

Appendix B. IAB Members at the Time of This Writing

Loa Andersson
Gonzalo Camarillo
Stuart Cheshire
Russ Housley
Olaf Kolkman
Gregory Lebovitz
Barry Leiba
Kurtis Lindqvist
Andrew Malis
Danny McPherson
David Oran
Dave Thaler
Lixia Zhang

Authors' Addresses

Bernard Aboba
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052

EMail: bernarda@microsoft.com

Dave Thaler
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052

EMail: dthaler@microsoft.com

Loa Andersson
Ericsson AB

EMail: loa.andersson@ericsson.com

Stuart Cheshire
Apple Computer, Inc.
1 Infinite Loop
Cupertino, CA 95014

EMail: cheshire@apple.com