Network Working Group
Internet-Draft
Intended status: Informational
Expires: December 31, 2018

C. Huitema
Private Octopus Inc.

June 29, 2018

DNS-SD Privacy and Security Requirements
draft-huitema-dnssd-prireq-00

Abstract

DNS-SD (DNS Service Discovery) normally discloses information about devices offering and requesting services. This information includes host names, network parameters, and possibly a further description of the corresponding service instance. Especially when mobile devices engage in DNS Service Discovery over Multicast DNS at a public hotspot, serious privacy problems arise. We analyze the requirements of a privacy respecting discovery service.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on December 31, 2018.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction

DNS-SD [RFC6763] over mDNS [RFC6762] enables zero-configuration service discovery in local networks. It is very convenient for users, but it requires the public exposure of the offering and requesting identities along with information about the offered and requested services. Parts of the published information can seriously breach the user's privacy. These privacy issues and potential solutions are discussed in [KW14a], [KW14b] and [K17].

There are cases when nodes connected to a network want to provide or consume services without exposing their identity to the other parties connected to the same network. Consider for example a traveler wanting to upload pictures from a phone to a laptop when connected to

the Wi-Fi network of an Internet cafe, or two travelers who want to share files between their laptops when waiting for their plane in an airport lounge.

We expect that these exchanges will start with a discovery procedure using DNS-SD [RFC6763] over mDNS [RFC6762]. One of the devices will publish the availability of a service, such as a picture library or a file store in our examples. The user of the other device will discover this service, and then connect to it.

When analyzing these scenarios in Section 3, we find that the DNS-SD messages leak identifying information such as the instance name, the host name or service properties.

1.1. Requirements

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

2. DNS-SD Discovery Scenarios

DNS-Based Service Discovery (DNS-SD) is defined in [RFC6763]. It allows nodes to publish the availability of an instance of a service by inserting specific records in the DNS ([RFC1033], [RFC1034], [RFC1035]) or by publishing these records locally using multicast DNS (mDNS) [RFC6762]. Available services are described using three types of records:

PTR Record: Associates a service type in the domain with an "instance" name of this service type.

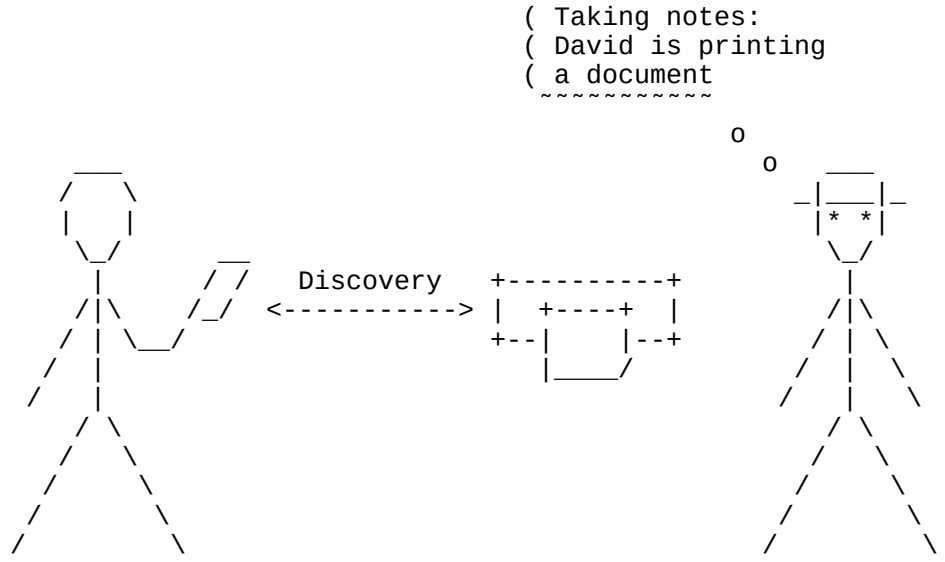
SRV Record: Provides the node name, port number, priority and weight associated with the service instance, in conformance with [RFC2782].

TXT Record: Provides a set of attribute-value pairs describing specific properties of the service instance.

In the remaining sections, we review common discovery scenarios provided by DNS-SD and discuss their privacy requirements.

2.1. Private client and public server

Perhaps the simplest private discovery scenario involves a single client connecting to a public server through a public network. A common example would be a traveler using a publicly available printer in a business center, in an hotel or at an airport.

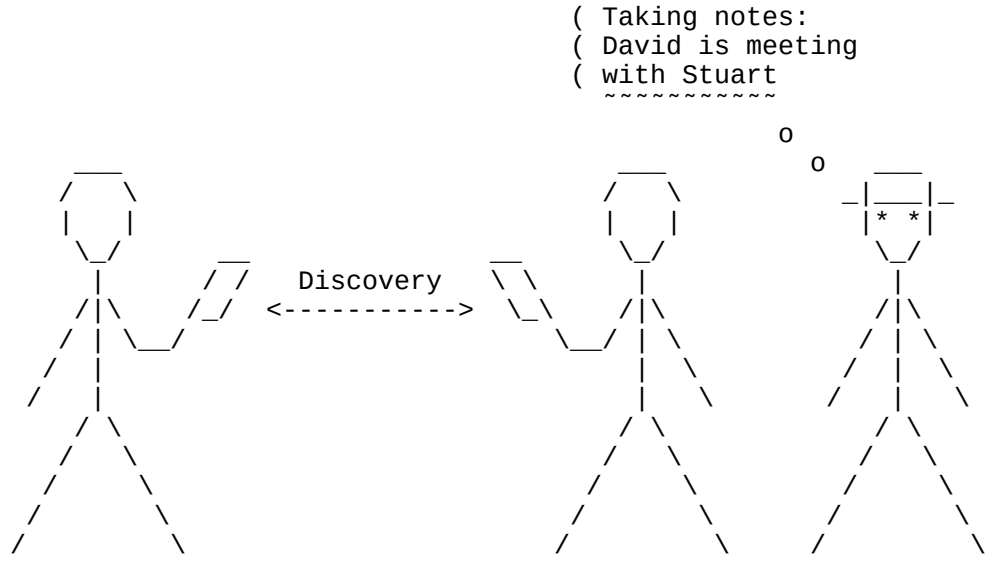


In that scenario, the server is public and wants to be discovered, but the client is private. The adversary will be listening to the network traffic, trying to identify the visitors' devices and their activity. Identifying devices leads to identifying people, either just for tracking people or as a preliminary to targeted attacks.

The requirement in that scenario is that the discovery activity should not disclose the identity of the client.

2.2. Private client and private server

The second private discovery scenario involves private client connecting to a private server. A common example would be two people engaging in a collaborative application in a public place, such as for example an airport's lounge.

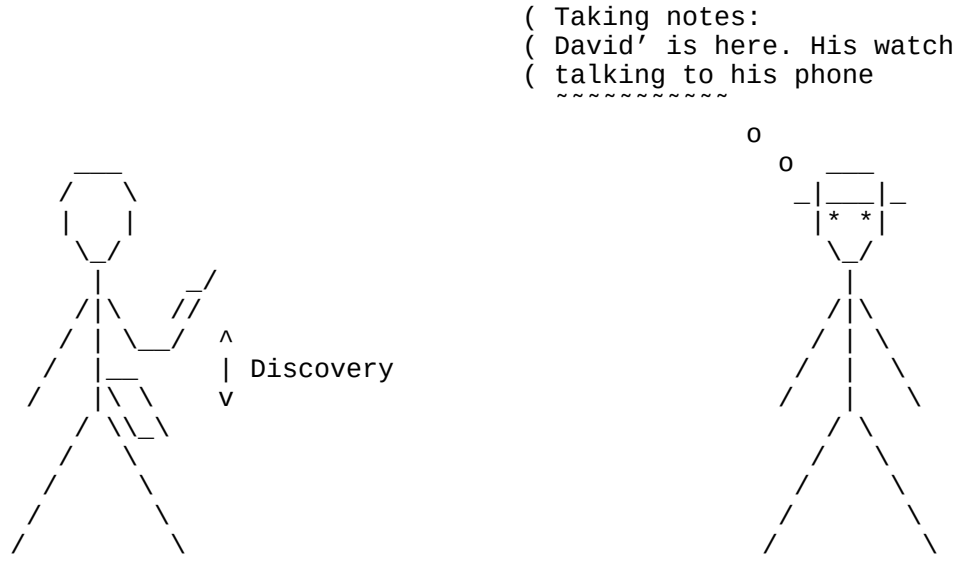


In that scenario, the collaborative application on one of the device will act as server, and the application on the other device will act as client. The server wants to be discovered by the client, but has no desire to be discovered by anyone else. The adversary will be listening to network traffic, attempting to discover the identity of devices as in the first scenario, and also attempting to discover the patterns of traffic, as these patterns reveal the business and social interactions between the owners of the devices.

The requirement in that scenario is that the discovery activity should not disclose the identity of either the client or the server.

2.3. Wearable client and server

The third private discovery scenario involves wearable devices. A typical example would be the watch on someone’s wrist connecting to the phone in their pocket.



This third scenario is in many ways similar to the second scenario. It involves two devices, one acting as server and the other acting as client, and it leads to the same requirement that the discovery traffic not disclose the identity of either the client or the server. The main difference is that the devices are managed by a single owner, which can lead to different methods for establishing secure relations between the device. There is also an added emphasis in hiding the type of devices that the person wears.

In addition to tracking the identity of the owner of the devices, the adversary is interested by the characteristics of the devices, such as type, brand, and model. Identifying the type of device can lead to further attacks, from theft to device specific hacking. The combination of devices worn by the same person will also provide a "fingerprint" of the person, allowing identification.

3. Privacy Considerations

The discovery scenarios in Section Section 2 illustrate three separate privacy requirements that vary based on use case:

1. Client identity privacy: Client identities are not leaked during service discovery or use.

2. Multi-owner, mutual client and server identity privacy: Neither client nor server identities are leaked during service discovery or use.

3. Single-owner, mutual client and server identity privacy: Identities of clients and servers owned and managed by the same application, device, or user are not leaked during service discovery or use.

In the remaining subsections, we describe aspects of DNS-SD that make these requirements difficult to achieve in practice.

3.1. Privacy Implication of Publishing Service Instance Names

In the first phase of discovery, client obtain all PTR records associated with a service type in a given naming domain. Each PTR record contains a Service Instance Name defined in Section 4 of [RFC6763]:

Service Instance Name = <Instance> . <Service> . <Domain>

The <Instance> portion of the Service Instance Name is meant to convey enough information for users of discovery clients to easily select the desired service instance. Nodes that use DNS-SD over mDNS [RFC6762] in a mobile environment will rely on the specificity of the instance name to identify the desired service instance. In our example of users wanting to upload pictures to a laptop in an Internet Cafe, the list of available service instances may look like:

```
Alice's Images          . _imageStore._tcp . local
Alice's Mobile Phone    . _presence._tcp   . local
Alice's Notebook        . _presence._tcp   . local
Bob's Notebook          . _presence._tcp   . local
Carol's Notebook        . _presence._tcp   . local
```

Alice will see the list on her phone and understand intuitively that she should pick the first item. The discovery will "just work".

However, DNS-SD/mDNS will reveal to anybody that Alice is currently visiting the Internet Cafe. It further discloses the fact that she uses two devices, shares an image store, and uses a chat application supporting the _presence protocol on both of her devices. She might currently chat with Bob or Carol, as they are also using a _presence supporting chat application. This information is not just available to devices actively browsing for and offering services, but to anybody passively listening to the network traffic.

3.2. Privacy Implication of Publishing Node Names

The SRV records contain the DNS name of the node publishing the service. Typical implementations construct this DNS name by concatenating the "host name" of the node with the name of the local

domain. The privacy implications of this practice are reviewed in [RFC8117]. Depending on naming practices, the host name is either a strong identifier of the device, or at a minimum a partial identifier. It enables tracking of both the device, and, by extension, the device's owner.

3.3. Privacy Implication of Publishing Service Attributes

The TXT record's attribute-value pairs contain information on the characteristics of the corresponding service instance. This in turn reveals information about the devices that publish services. The amount of information varies widely with the particular service and its implementation:

- Some attributes like the paper size available in a printer, are the same on many devices, and thus only provide limited information to a tracker.

- Attributes that have freeform values, such as the name of a directory, may reveal much more information.

Combinations of attributes have more information power than specific attributes, and can potentially be used for "fingerprinting" a specific device.

Information contained in TXT records does not only breach privacy by making devices trackable, but might directly contain private information about the user. For instance the _presence service reveals the "chat status" to everyone in the same network. Users might not be aware of that.

Further, TXT records often contain version information about services allowing potential attackers to identify devices running exploit-prone versions of a certain service.

3.4. Device Fingerprinting

The combination of information published in DNS-SD has the potential to provide a "fingerprint" of a specific device. Such information includes:

- List of services published by the device, which can be retrieved because the SRV records will point to the same host name.

- Specific attributes describing these services.

- Port numbers used by the services.

o Priority and weight attributes in the SRV records.

This combination of services and attributes will often be sufficient to identify the version of the software running on a device. If a device publishes many services with rich sets of attributes, the combination may be sufficient to identify the specific device.

A sometimes heard argument is that devices providing services can be identified by observing the local traffic, and that trying to hide the presence of the service is futile. This argument, however, does not carry much weight because

1. Proving privacy at the discovery layer is of the essence for enabling automatically configured privacy-preserving network applications. Application layer protocols are not forced to leverage the offered privacy, but if device tracking is not prevented at the deeper layers, including the service discovery layer, obfuscating a certain service's protocol at the application layer is futile.

2. Further, even if the application layer does not protect privacy, it is hard to record and analyse the unicast traffic (which most applications will generate) compared to just listening to the multicast messages sent by DNS-SD/mDNS.

The same argument can be extended to say that the pattern of services offered by a device allows for fingerprinting the device. This may or may not be true, since we can expect that services will be designed or updated to avoid leaking fingerprints. In any case, the design of the discovery service should avoid making a bad situation worse, and should as much as possible avoid providing new fingerprinting information.

3.5. Privacy Implication of Discovering Services

The consumers of services engage in discovery, and in doing so reveal some information such as the list of services they are interested in and the domains in which they are looking for the services. When the clients select specific instances of services, they reveal their preference for these instances. This can be benign if the service type is very common, but it could be more problematic for sensitive services, such as for example some private messaging services.

One way to protect clients would be to somehow encrypt the requested service types. Of course, just as we noted in Section 3.4, traffic analysis can often reveal the service.

4. Security Considerations

For each of the operations described above, we must also consider security threats we are concerned about.

4.1. Authenticity, Integrity & Freshness

Can we trust the information we receive? Has it been modified in flight by an adversary? Do we trust the source of the information? Is the source of information fresh, i.e., not replayed? Freshness may or may not be required depending on whether the discovery process is meant to be online. In some cases, publishing discovery information to a shared directory or registry, rather than to each online recipient through a broadcast channel, may suffice.

4.2. Confidentiality

Confidentiality is about restricting information access to only authorized individuals. Ideally this should only be the appropriate trusted parties, though it can be challenging to define who are "the appropriate trusted parties." In some uses cases, this may mean that only mutually authenticated and trusting clients and servers can read messages sent for one another. The "Discover" operation in particular is often used to discover new entities that the device did not previously know about. It may be tricky to work out how a device can have an established trust relationship with a new entity it has never previously communicated with.

4.3. Resistance to Dictionary Attacks

It can be tempting to use (publicly computable) hash functions to obscure sensitive identifiers. This transforms a sensitive unique identifier such as an email address into a "scrambled" (but still unique) identifier. Unfortunately simple solutions may be vulnerable to offline dictionary attacks.

4.4. Resistance to Denial-of-Service Attack

In any protocol where the receiver of messages has to perform cryptographic operations on those messages, there is a risk of a brute-force flooding attack causing the receiver to expend excessive amounts of CPU time (and battery power) just processing and discarding those messages.

4.5. Resistance to Sender Impersonation

Sender impersonation is an attack wherein messages such as service offers are forged by entities who do not possess the corresponding secret key material. These attacks may be used to learn the identity of a communicating party, actively or passively.

4.6. Sender Deniability

Deniability of sender activity, e.g., of broadcasting a discovery request, may be desirable or necessary in some use cases. This property ensures that eavesdroppers cannot prove senders issued a specific message destined for one or more peers.

5. Operational Considerations

5.1. Power Management

Many modern devices, especially battery-powered devices, use power management techniques to conserve energy. One such technique is for a device to transfer information about itself to a proxy, which will act on behalf of the device for some functions, while the device itself goes to sleep to reduce power consumption. When the proxy determines that some action is required which only the device itself can perform, the proxy may have some way (such as Ethernet "Magic Packet") to wake the device.

In many cases, the device may not trust the network proxy sufficiently to share all its confidential key material with the proxy. This poses challenges for combining private discovery that relies on per-query cryptographic operations, with energy-saving techniques that rely on having (somewhat untrusted) network proxies answer queries on behalf of sleeping devices.

5.2. Protocol Efficiency

Creating a discovery protocol that has the desired security properties may result in a design that is not efficient. To perform the necessary operations the protocol may need to send and receive a large number of network packets. This may consume an unreasonable amount of network capacity (particularly problematic when it’s shared wireless spectrum), cause an unnecessary level of power consumption (particularly problematic on battery devices) and may result in the discovery process being slow.

It is a difficult challenge to design a discovery protocol that has the property of obscuring the details of what it is doing from

unauthorized observers, while also managing to do that quickly and efficiently.

5.3. Secure Initialization and Trust Models

One of the challenges implicit in the preceding discussions is that whenever we discuss "trusted entities" versus "untrusted entities", there needs to be some way that trust is initially established, to convert an "untrusted entity" into a "trusted entity".

One way to establish trust between two entities is to trust a third party to make that determination for us. For example, the X.509 certificates used by TLS and HTTPS web browsing are based on the model of trusting a third party to tell us who to trust. There are some difficulties in using this model for establishing trust for service discovery uses. If we want to print our tax returns or medical documents on "our" printer, then we need to know which printer on the network we can trust be be "our" printer. All of the printers we discover on the network may be legitimate printers made by legitimate printer manufacturers, but not all of them are "our" printer. A third-party certificate authority cannot tell us which one of the printers is ours.

Another common way to establish a trust relationship is Trust On First Use (TOFU), as used by ssh. The first usage is a Leap Of Faith, but after that public keys are exchanged and at least we can confirm that subsequent communications are with the same entity. In today’s world, where there may be attackers present even at that first use, it would be preferable to be able to establish a trust relationship without requiring an initial Leap Of Faith.

Techniques now exist for securely establishing a trust relationship without requiring an initial Leap Of Faith. Trust can be established securely using a short passphrase or PIN with cryptographic algorithms such as Secure Remote Password (SRP) [RFC5054] or a Password Authenticated Key Exchange like J-PAKE [RFC8236] using a Schnorr Non-interactive Zero-Knowledge Proof [RFC8235].

Such techniques require a user to enter the correct passphrase or PIN in order for the cryptographic algorithms to establish working communication. This avoids the human tendency to simply press the "OK" button when asked if they want to do something on their electronic device. It removes the human fallibility element from the equation, and avoids the human users inadvertently sabotaging their own security.

Using these techniques, if a user tries to print their tax return on a printer they’ve never used before (even though the name looks

right) they'll be prompted to enter a pairing PIN, and the user *cannot* ignore that warning. They can't just press an "OK" button. They have to walk to the printer and read the displayed PIN and enter it. And if the intended printer is not displaying a pairing PIN, or is displaying a different pairing PIN, that means the user may be being spoofed, and the connection will not succeed, and the failure will not reveal any secret information to the attacker. As much as the human desires to "just give me an OK button to make it print" (and the attacker desires them to click that OK button too) the cryptographic algorithms do not give the user the ability to opt out of the security, and consequently do not give the attacker any way to persuade the user to opt out of the security protections.

5.4. External Dependencies

Trust establishment may depend on external, and optionally online, parties. Systems which have such a dependency may be attacked by interfering with communication to external dependencies. Where possible, such dependencies should be minimized. Local trust models are best for secure initialization in the presence of active attackers.

6. IANA Considerations

This draft does not require any IANA action.

7. Acknowledgments

This draft incorporates many contributions from Stuart Cheshire and Chris Wood.

8. Informative References

[K17] Kaiser, D., "Efficient Privacy-Preserving Configurationless Service Discovery Supporting Multi-Link Networks", 2017, <http://nbn-resolving.de/urn:nbn:de:bsz:352-0-422757>.

[KW14a] Kaiser, D. and M. Waldvogel, "Adding Privacy to Multicast DNS Service Discovery", DOI 10.1109/TrustCom.2014.107, 2014, <http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=7011331>.

[KW14b] Kaiser, D. and M. Waldvogel, "Efficient Privacy Preserving Multicast DNS Service Discovery", DOI 10.1109/HPCC.2014.141, 2014, <http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=7056899>.

[RFC1033] Lottor, M., "Domain Administrators Operations Guide", RFC 1033, DOI 10.17487/RFC1033, November 1987, <https://www.rfc-editor.org/info/rfc1033>.

[RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, DOI 10.17487/RFC1034, November 1987, <https://www.rfc-editor.org/info/rfc1034>.

[RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <https://www.rfc-editor.org/info/rfc1035>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC2782] Gulbrandsen, A., Vixie, P., and L. Esibov, "A DNS RR for specifying the location of services (DNS SRV)", RFC 2782, DOI 10.17487/RFC2782, February 2000, <https://www.rfc-editor.org/info/rfc2782>.

[RFC5054] Taylor, D., Wu, T., Mavrogiannopoulos, N., and T. Perrin, "Using the Secure Remote Password (SRP) Protocol for TLS Authentication", RFC 5054, DOI 10.17487/RFC5054, November 2007, <https://www.rfc-editor.org/info/rfc5054>.

[RFC6762] Cheshire, S. and M. Krochmal, "Multicast DNS", RFC 6762, DOI 10.17487/RFC6762, February 2013, <https://www.rfc-editor.org/info/rfc6762>.

[RFC6763] Cheshire, S. and M. Krochmal, "DNS-Based Service Discovery", RFC 6763, DOI 10.17487/RFC6763, February 2013, <https://www.rfc-editor.org/info/rfc6763>.

[RFC8117] Huitema, C., Thaler, D., and R. Winter, "Current Hostname Practice Considered Harmful", RFC 8117, DOI 10.17487/RFC8117, March 2017, <https://www.rfc-editor.org/info/rfc8117>.

[RFC8235] Hao, F., Ed., "Schnorr Non-interactive Zero-Knowledge Proof", RFC 8235, DOI 10.17487/RFC8235, September 2017, <https://www.rfc-editor.org/info/rfc8235>.

[RFC8236] Hao, F., Ed., "J-PAKE: Password-Authenticated Key Exchange by Juggling", RFC 8236, DOI 10.17487/RFC8236, September 2017, <https://www.rfc-editor.org/info/rfc8236>.

Author's Address

Christian Huitema
Private Octopus Inc.
Friday Harbor, WA 98250
U.S.A.

Email: huitema@huitema.net
URI: http://privateoctopus.com/