GDP Discovery Service (GDPDS)

December 5th, 2016

Abstract

The GDP Discovery Service (GDPDS) provides secure discovery and advertisement to clients using the Global Data Plane (GDP). GDPDS servers advertise themselves on local networks, allowing clients to connect to them and advertise their capabilities to other clients connected to the GDP. GDPDS servers share information about clients using a distributed hash table (DHT). A global registry serves as a search engine which accesses the discovery DHT and provides an API to applications for searching for clients. This allows clients to identify clients which they have permission to interact with and which possess specific capabilities. GDPDS is designed to be used by lightweight and mobile clients while still providing the option of client authentication.

Introduction

The Global Data Plane (GDP) provides a flat name space through which clients can communicate using logs. However, without a discovery service, GDP log names and client GUIDs must be obtained out of band. GDPDS provides a system through which clients can identify the log names and GUIDs of other clients which they can communicate with via GDP logs.

Many IoT devices using the GDP may have limited computation and communication capabilities. This presents a requirement that GDPDS be designed to minimize data traffic and client-side computation. At odds with this requirement; however, is the importance of authenticating devices. In order to verify clients' identities as they register with the discovery service, the clients must be capable of using RSA. In order to strike a balance between these competing requirements, the GDPDS is designed to minimize communication with the discovery service. Clients must still use RSA algorithms in order to authenticate themselves, though this is made optional so low-power devices still have the potential to use the service.

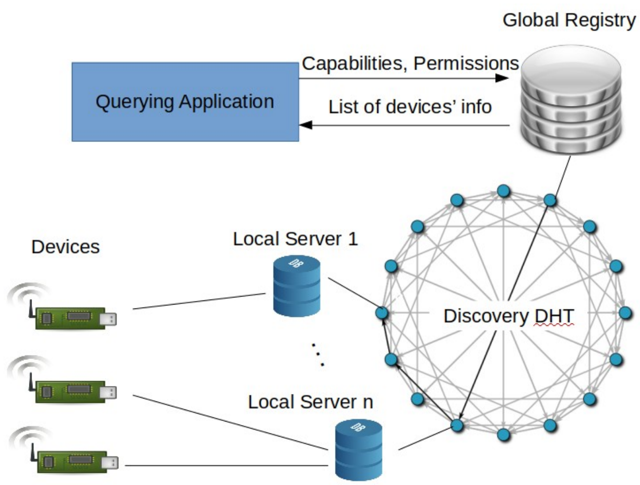
Overview of GDPDS

In order to accommodate as wide of a variety of client devices as possible, GDPDS is designed to minimize data transmission over networks and client-side computation.

Clients' information is stored permanently within an "info log" in the GDP so clients need only communicate log names and their GUID to discovery servers rather than needing to send them potentially lengthy lists of their capabilities and allowable permissions. The discovery server then reads the client's corresponding info log and stores its information within a MySQL database and a DHT. This local database stores client information and can be queried directly for information about clients on the local subnet.

The DHT is shared between local discovery servers on different networks, allowing clients anywhere to be identified. GDPDS includes a global registry server which provides an API to querying for devices based on capability and permission.

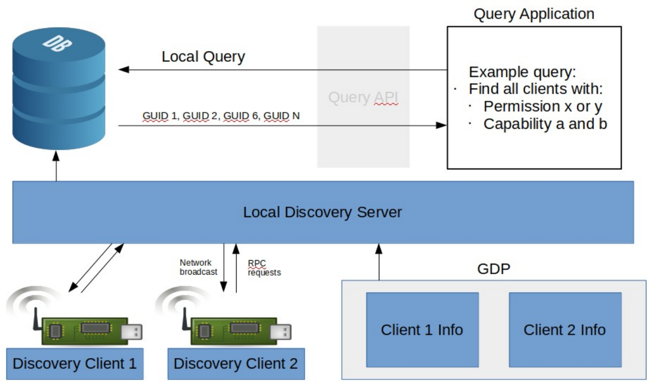
GDPDS has also been designed with the consideration that the discovery service needs to verify that the clients which are advertising themselves to GDPDS servers are the clients they claim to be and possess the capabilities they claim to have. This requirement has been achieved using RSA authentication, linking clients to info logs and the ability to verify the validity of info logs by verifying the chain of trust within the certificate each info log has.



The primary contents of the discovery package are the "global_registry," "local_server" and "client" modules. To ease the process of preparing info logs and querying discovery databases for client information, "client_setup" and "query" modules have also been included within the discovery package. The "dht_service" module is used internally by the local server and global registry. The purpose of each of these modules is summarized below:

- **global_registry:**
  The global registry provides a restful API which allows info logs, which describe a class of devices with the same capabilities and permissions, to be registered. It also connects with the DHT and returns search results for querying devices based on their capabilities, permissions and activity.
- **local_server:**
  The local server module advertises its presence on a local network using Avahi. It accepts client registration requests, reads clients' "info logs," and logs their information within a MySQL database.

- **client:**
  The client module searches for discovery servers which are broadcasting their presence on a network and sends them a connection request. Once registered with a local discovery server, the client sends periodic renewals to avoid timing out with the discovery service.
- **client_setup:**
  The client_setup module writes a set of capabilities, permissions, and a certificate to an info log. It also generates a client key pair, writing the public key to the info log. Since info logs can describe multiple clients with the same allowable permissions and capabilities, this module can generate multiple client key pairs for a given info log.
- **local_query:**
  The query module provides an interface for applications to query a discovery database for clients based on their capabilities and allowable permissions.
- **dht_service:**
  The DHT service provides a restful API to other GDPDS modules for accessing the discovery DHT.



Implementation

Each client using the GDPDS must have an info log which describes it, and optionally may have an output log and input log which can be used to communicate with other clients via the GDP. When advertising itself to a discovery server, the client must communicate each of its corresponding log names using the GDPDS RPC interface. The discovery local server can then read the client's info log, authenticate the client and its info log, log the client's information in its database and update the discovery DHT with the client's information.

Local Discovery Servers

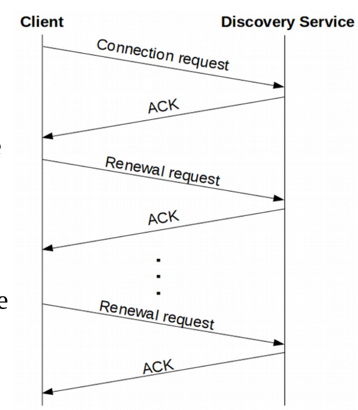
GDPDS is designed around the idea that clients can best be identified by their GUID, capabilities and allowable permissions. Capabilities, represented as strings, allow for the GDPDS to be used with any ontology. While GUIDs and capabilities are relatively straight forward, the concept of permissions requires more explanation. If clients were queried based solely on capabilities, the query could return many clients which the querier does not have the ability to communicate with because it does not have keys to decrypt information in the queried clients' output logs or does not have the ability to write to clients' input logs. Permissions provide a way for queried clients to pruned based on whether or not a querier has permission to interact with them. Therefore, info logs contain a list of permissions, which, like capabilities, can be any string. A querier can specify which permission strings returned clients must have in order to ensure that the the querier has the ability to interact with the returned clients.

Local discovery databases are designed to be ephemeral, storing only the data of clients currently connected to them. Databases can be started and stopped, requiring only a new connection signal from clients to re-log them in their databases. Discovery databases store client information in three MySQL tables:

- **clients:**
  Each entry contains a client's guid, IP address, info log name, input log name, output log name, authentication status and certification status
- **capabilities:**
  Each entry contains a 1 to 1 mapping of a client GUID and a capability
- **permissions:**
  Each entry contains a 1 to 1 mapping of a client GUID and a permission

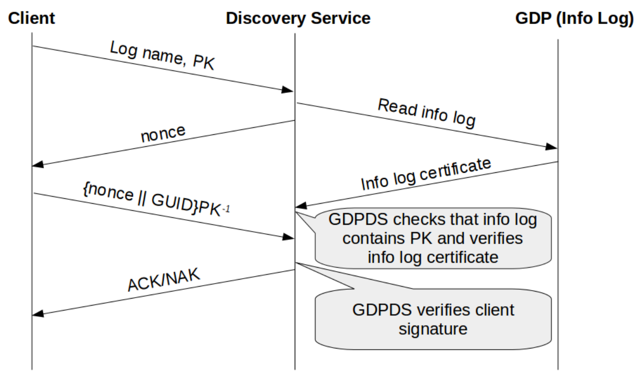
RPC interface

The RPC interface used by the GDPDS is both general and lightweight. Each RPC request contains a field specifying the length of the request name, the type of RPC message (request, ACK or NAK) and the payload length. The first field of the payload must be the request name, allowing the discovery server to dispatch the request to the proper handler.



Since many clients may be mobile, a renewal request must be sent to the discovery service at least once ever 30 seconds. If a renewal is not sent within that time period, the discovery server will delete the client from its database and a new connection request must be sent for the client to re-egister.

Security



To prevent attacks where clients impersonate other clients or deceive querying clients into thinking the attacker has capabilities or permissions it does not, GDPDS provides the option for the clients to authenticate themselves and their info logs.

Client authentication creates a verifiable one-to-many mapping between an info log and one or more clients. Since the info log contains the public key of each of the clients it represents, and each client contains its own private key, the discovery server can verify that (a) the client's public key is in the info log and (b) the client possesses the corresponding private key. Client_setup currently uses 2048-bit RSA keys.

A certificate embedded in an info log creates a chain of trust which verifies the validity of an info log's contents. Info log certificates are X509 certificates which are signed using SHA-256 by a certificate authority during initial info log setup.

While client authentication and info log certificates are optional, the discovery database stores whether or not clients are authenticated and/or certified, so querying clients can prune results which are potentially not authentic. Client authentication and certificates were made optional so low power devices do not have to support RSA computation and to reduce the setup complexity created by the need for certificate authorities.

Discovery DHT

In order to make information about clients and their location available globally, local servers put a new entry into a shared DHT each time they receive a connection or renewal from a client.

The Discovery DHT stores mappings of an info log name as a key and a list of clients' metadata as a value. For each client in the list corresponding to a info log, the client's GUID, time of last connection / renewal, IP address of logging local server, input log and output log is stored. Entries in the DHT are removed automatically after 5 minutes to avoid unnecessary storage of data for inactive clients.

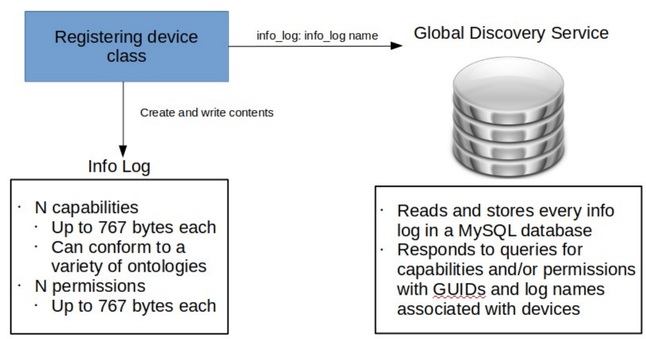
In order for local servers to connect to the DHT, they must be given the addresss and bootstrap port of an existing node in the DHT. Since there may be only one global registry (as described below), it is assumed to be the bootstrap node for new local servers joining the network.

Global Registry

The global registry runs an HTTP server which provides a restful API for registering new device classes with the registry and for querying for active clients.

The process for registering a device class simply requires creating an info log and writing the devices' capabilities and permissions to it using the client setup module. Lastly, the name of the info log must be sent as an HTTP PUT to the global registry.

The global registry stores device class information it reads from the info logs corresponding to newly registered device classes in a MySQL database. Similar to the database used in the local servers, the database consists of a permissions and a capabilities table which map info log names to permissions and capabilities respectively.



The global registry handles HTTP GET requests which query for devices by capabilities, permissions and last activity time using a two step filtering process:
1. The registry's MySQL database is searched for all info logs which contain all of the specified capabilities and at least one of the specified permissions. If "any" capabilities are specified, info logs will only be filtered by permissions.
2. The registry gets all of the devices logged in the Discovery DHT for each info log found in (1). Of the devices found in the DHT, only those which have datetimes after the 'active_since' query parameter are returned.

Future Work

Some of the potential extensions of GDPDS are described below:

- **UI for Global Registry:**
  A web UI for the global registry would allow users to type capability, permission and activity information in user-friendly boxes and see a table of results. A screen for registering new device classes could also be included.

- **Global Registry optimization:**
  Searches in the Global Registry's MySQL database are O(n) in the current implementation. This search time could be reduced by using hashing and a hierarchical ontology for capabilities.
- **Lighter weight client module:**
  The client module could be implemented in C to allow it to run more efficiently on devices with limited resources.
- **Caching of client data**:
  Local servers could be augmented to include a cache, which stores info log contents so re-connecting clients would not have to wait for the local server to read its info log each time it wishes to advertise itself.
- **Ontology:**
  GDPDS is a platform which can use any ontology; however, choosing one and layering it on top of GDPDS could improve its usability and allow for further optimization.
- **Improved security:**
  Devices are authenticated at the local server level, however, there are limited security features in place to prevent malicious servers from connecting to the Discovery DHT and storing incorrect information in it. A system for authenticating local servers could be implemented

Conclusions

GDPDS uses a novel, scalable system which enables mobile devices to advertise their location, capabilities and permissions. Through GDPDS's Global Registry, it exposes an API which allows devices to be searched by their capabilities, permissions and activity globally.

GDPDS utilizes the Global Data Plane by using info logs to store information about clients in a way which is ontology-independent allows for easy queries. This system shows how large amounts of information about devices can be obtained without devices needing to communicate it themselves.

Additionally, GDPDS utilizes an authentication system uniquely suited for IoT applications where it is important to verify device identity and capabilities. A major downside of this system; however, is the requirement that devices be capable of using RSA.