Conceptual API for ASAs using GDNP

Draft 2015-05-14
Brian Carpenter

GDNP (Generic Discovery and Negotiation Protocol) is a proposed approach to the signaling protocol for autonomic networking (see https://tools.ietf.org/html/draft-carpenter-anima-gdn-protocol).

The assumption of this document is that any Autonomic Service Agent (ASA) needs to call a GDNP module that handles protocol details (security, sending and listening for GDNP messages, waiting, caching discovery results, negotiation looping, sending and receiving sychronization data, etc.) but understands nothing about individual objectives. So this is a high level abstract API for use by ASAs. In theory, an ASA designer would not need to know protocol details of GDNP.

Because the GDNP protocol includes wait states, and the ASA itself may include wait states, the GDNP module will need to be asynchronous with the ASA. In this abstract API, that is handled by providing two types of calls: GDNP functions named `gdn_XXX` that may be called by each ASA, and functions provided by each ASA that are named `asa_XXX`. A thread in the ASA may call `gdn_XXX` functions, and a thread in the GDNP module may call `asa_XXX` functions.

This approach delegates listen states, the negotiation loop and most wait states to the GDNP module, simplifying things for the ASA. There could be better way to model the process without asynchronous calls – ideas welcome!

The design makes an assumption that whatever the format of a negotiation objective, two values of the objective may be compared to determine whether value A is considered greater than or equal to value B. The ASA provides the comparison function, which may or may not be an arithmetic comparison. For example there could be an ASA that considers 'lemons' to be less than 'oranges', and 'apples' to be greater than or equal to 'oranges' but less than 'bananas'. Thus, if the acceptable range of values in a negotiation was stated as 'oranges' to 'bananas', 'apples' would be in range but 'lemons' would not.

The design allows for the rapid mode defined in GDNP, where a discovery message immediately triggers negotiation or synchronization. This is a bit complicated and we should consider whether it's justified.

First we list the functions in the conceptual API, then a few ABNF definitions. There are forward references, so this needs to be read at least twice to understand it...

Functions:

```
gdn_init(objective_list, *asa_result) -> OK/fail
```

This initialises state in the GDNP module for the calling entity (the ASA). The `objective_list` parameter is a list of all GDNP objective options that the ASA supports (see below for ABNF and more details). The `*asa_result` parameter is a reference to a function within the ASA that GDNP may call asynchronously, described later.

We assume here that `gdn_init` invokes the security bootstrap procedure for the calling ASA if necessary. That is a big assumption.

We also assume that `gdn_init` records the list of objectives that the ASA is able to support for discovery, negotiation, and/or synchronization. The flags included in the formal definition of `objective` below should make this clearer.

Thus, a successful call to `gdn_init` means that the GDNP module now has security credentials for the ASA and is aware of all the objectives that the ASA can handle.

`OK`: means that the ASA has been authenticated and has credentials to proceed.

`fail`: means that the ASA has not been authenticated and cannot operate.

`gdn_discover(objective) -> OK/fail`

This causes GDNP discovery for the listed objective.

`OK`: means that discovery has succeeded and the locator has been cached by GDNP.

`fail`: means that discovery has failed. The ASA must wait before retrying. The retry timeout is a parameter of the ASA.

See below for the case of rapid-mode negotiation or synchronization triggered via discovery.

Open issue: what happens when GDNP receives multiple valid discovery responses?

`gdn_negotiate(objective) -> objective/fail`

This causes GDNP to start negotiation for the given objective, and to negotiate within the limits set in the objective (see below).

`objective`: Negotiation succeeded, the return parameter `objective` contains the negotiated `value`.

`fail`: Negotiation failed. The ASA must wait before retrying. The retry timeout is a parameter of the ASA.

`gdn_synchronize(objective) -> objective/fail`

This sends a GDNP synchronization request for the given objective.

`objective`: Synchronization succeeded, the return parameter `objective` contains the received `value`.

`fail`: Synchronization failed. The ASA must wait before retrying. The retry timeout is a parameter of the ASA.

`gdn_listen_negotiate(objective) -> OK/fail`

This causes GDNP to listen for negotiation requests for the given objective, and to negotiate within the limits set (see below). This call may be repeated whenever the limits change.

`OK`: GDNP is listening.

`fail`: GDNP is not listening.

`gdn_stop_negotiate(objective) -> OK/fail`

This causes GDNP to stop listening for negotiation requests for the given objective.

`OK`: GDNP is not listening.

`fail`: unspecified failure.

`gdn_listen_synchronize(objective) -> OK/fail`

This causes GDNP to listen for synchronization requests for the given objective, and to reply with the given value. This call may be repeated whenever the value changes.

`OK`: GDNP is listening.

`fail`: GDN is not listening.

`gdn_stop_synchronize(objective) -> OK/fail`

This causes GDNP to stop listening for synchronization requests for the given objective.

`OK`: GDNP is not listening.

`fail`: unspecified failure.

`asa_result(objective) -> objective/fail`

This is a function in the ASA called asynchronously by GDNP, if the ASA has previously called `gdn_listen_negotiate` and an incoming negotiation has terminated. It is also called after rapid-mode negotiation or synchronization via `gdn_discover`.

`objective`: Operation succeeded, this contains the negotiated or synchronized `value`.

`fail`: Operation failed. (TBD: need to indicate *which* operation failed.)

`asa_geq(objective_id value value) -> true/false`

This function is optionally provided by the ASA for use by the GDNP module. It returns true iff the first value is considered greater than or equal to the second value. The semantics of this comparison are known only to the ASA; the GDNP module uses this function to conduct negotiation to find an acceptable value between the set limits. If no function is provided, GDNP uses normal arithmetic comparison of the values, considered as unsigned binary numbers.

ABNF definitions:

```
objective_list = 1*(objective ["*asa_geq"])

objective = objective_id [D_flag] [N_flag] [S_flag] [value [limits]]
[loop_count] [timeout]

objective_id = <identifier for this objective>

;In the current GDNP design, this is an IANA-assigned option number

D_flag = true/false

; If this flag is set in an objective included in gdnp_init, it indicates that the
; calling ASA is willing to respond to discovery messages for that particular objective.
; In gdn_discover it indicates that discovery is requested.
```

N_flag = true/false

; If this flag is set in an objective included in gdnp_init, it indicates that the
; calling ASA is able to negotiate values for that particular objective.
; In gdn_discover it indicates that rapid-mode negotiation is requested and
; other parameters follow.

S_flag = true/false

; If this flag is set in an objective included in gdnp_init, it indicates that the
; calling ASA is able to synchronize values for that particular objective.
; In gdn_discover it indicates that rapid-mode synchronization is requested.

value = <opaque variable-length field which only has meaning to the ASA>

limits = value value ; low and high values allowed in negotiation

loop_count = <decimal integer limit for negotiation loop>

; This is optional because GDNP has a specified default value

timeout = <decimal integer timeout for discovery or negotiation in milliseconds>

; This is optional because GDNP has a specified default value