Discovery for BRSKI

draft-eckert-anima-brski-discovery-05

IETF121 Dublin

Toerless Eckert, Futurewei USA (tte@cs.fau.de)

Esko Dijk (esko.dijk@iotconsultancy.nl)

11/05/2024

Vote for BRSKI today

Background

As needed (e.g. For DNS-SD review)

Requirements (1)

- BRSKI pledges need to discover+select a BRSKI registrar
  - May use TCP/TLS, COAPS – or in future other “transport” stacks (QUIC ?)
- Pledges may not have full network connectivity, but only L2
  - Requires use of TCP or UDP connection proxy
  - Pledge discovers+select proxy, Proxy discovers+selects Registrar
- Variation of BRSKI protocol
  - Preferences (pledge/registrar/backend) for pre-existing encoding library/method
    - vformat: cms, jose, cose, ...
  - Preferences for different backend credentials (CERT+TA) managemenet protocol
    - enroll: est, cmp, scep, ...
  - Difference in use-case workflow of messages/functionality
    - mode: rrm, prm
  - Should be extensible to more variation criteria
- Combinatorial
  - mode * vformat * enroll – one option for each of the three needed in every basic instantiation of BRSKI
  - Not all options supported with all transports. jose (currently only TCP), cose (currently) only COAPS
  - Only “requird by target deployemnt” combinations currently specified / understood
    - Analysis if a new variation will work out of the box or needs extension may be simple .. difficult

Requirements (2)

- Different deployments may prefer/require different discovery mechanisms
  - IoT solutions typically come with pre-set discovery method. New protocol/application should not require a different one
  - DNS-SD with mDNS and/or unicast-DNS
  - GRASP (from ANIMA-WG itself)
  - CORE-LF (from CORE-WG, used most often with COAP stacks/deployment, but not limited to that) L2-multicast/unicast
  - Future discovery methods ... ?!
- BRSKI Agents (or other devices) need to discover pledges
  - By some HW-identity derived name – across vendors
  - Browsing and selective
- Future discovery of other BRSKI roles
  - Registrar to discover MASA

Solution approach

- Create IANA registry (multiple tables) for required discovery data / code-points
  - Scatter/Gather implementations from variety of RFC otherwise error-prone/difficult
- Do not allocate new “service name” for every protocol variation
  - Would prohibit automatic support of new variations in proxies across arbitrary discovery methods
  - Instead, define single-string encoding of variations
    - And define for each discovery method an encoding for a list of such strings
- Allocate / register the necessary “service names” for every context
  - Whatever the equivalent for service name in DNS-SD are in other discovery mechanism
    - GRASP: objective name, CORE-LR: resource type
  - Role = Registrar, Pledge, MASA, ...
  - Context = Role plus Proxy for Role if applicable for specific transport
    - Different transports are incompatible and may have different variations
    - Need different service names for Role and Proxy-for-Role
  - Reuse same “service names” across context if possible
    - Can not e.g.: use same service name in DNS-SD to distinguish TLS, COAP and QUIC transport (2 * UDP!)
    - Solution: Add variation

BRSKI Variation Contexts

Context	Applicable Variation Types	Discovery Mechanism	Service Name(s) / Transport	Reference(s)
BRSKI	mode vformat enroll	GRASP	"AN_join_registrar" / "AN_Proxy" with IPPROTO_TCP	[RFC8995]
		DNS-SD	"brski-registrar" / "brski-proxy" with TCP	[RFC8995]
cBRSKI	mode vformat enroll	CORE-LF	rt=brski.jp with coaps	[I-D.ietf-anima-constrained-voucher]
			rt=brski.rs / rt=brski.rjpy with coaps	[I-D.ietf-anima-constrained-join-proxy]
		DNS-SD	"brski-proxy" / "brski-registrar" / "brski-registrar-rpy" with UDP	[THIS-RFC]
		GRASP	"AN_join_registrar" / "AN_join_registrar_rjp" / "AN_Proxy" with IPPROTO_UDP	[THIS-RFC]
BRSKI-PLEDGE	mode vformat enroll	DNS-SD	"brski-pledge" with TCP	[THIS-RFC]

BRSKI Variation Type and Choices Tabe

Context	Variation Type	Variation Type Choice	Reference	Flags	Note(s)
BRSKI, cBRSKI	mode	rrm	[RFC8995] ThisRFC	Dflt	Registrar Responder Mode the mode specified in [RFC8995]
		prm	ThisRFC		Pledge Responder Mode [I-D.ietf-anima-brski-prm]
BRSKI	vformat	cms	[RFC8368] ThisRFC	Dflt	CMS-signed JSON Voucher
		cose	ThisRFC		CBOR with COSE signature
cBRSKI		cose	ThisRFC	Dflt	CBOR with COSE signature [I-D.ietf-anima-constrained-voucher]
		cms	[RFC8368] ThisRFC		CMS-signed JSON Voucher

BRSKI Variation Strings Table

Context	Reference	Variation String	Variations	Explanations / Notes
BRSKI	[RFC8995]	"" / "EST-TLS"	rrm cms est	Note 1
	[I-D.ietf-anima-brski-ae]	cmp	rrm cms cmp	
	[I-D.ietf-anima-brski-prm]	prm-jose	prm jose est	
cBRSKI	[I-D.ietf-anima-constrained-voucher]	"" / "rrm-cose"	rrm cose est	

- When adding new variation types later, pre-existing variation strings will not include any choice for them. The “default”choice is therefore assumed to represent what the prior variations use.
- Note 1: The Variation String "EST-TLS" is equivalent to the Variation String "" and is required and only permitted for the AN_join_registrar objective value in GRASP for backward compatibility with RFC8995, where it is used for this variation. Note that AN_proxy uses "".
- *Only registering combinations actually used by reference specs! ... Future entries may not require RFC but could use other specs ... Up for us when deciding on Registry criterias. Currently spec required + expert review*

Summary of registry purpose / benefit

- Single place to find parameters for BRSKI discovery
  - “Service Names” for all discoverable BRSKI services across all supportable discovery mechanisms
  - Which spec defines which service / parameter / variation
- Composition and definition of variations
  - Simple variation strings registered on-demand when variation is needed and understood that it can work
- Extensible
  - Additional BRSKI services, discovery mechanisms and variation types easily added (and fully backward compatible)

Changes since IETF120

Changes since IETF120

- Title: “Discovery 4 Variations” -> “Discovery and Variations”
  - A lot what this does is applicable even without variations
- Overview text improved /expanded (no new technical content)
  - Challenges (variations, support for various discovery, agnostic proxies,..)
  - Functional summary
- Data Model / Registries text improved (no new technical content)
  - Variations needed because “service names” are not structured/hierarchical
  - Hopefully easier understandable text (and explanations shorter/better)
  - *Added note about ability to support QUIC across BRSKI proxies*

- Coalesced/New text for actual discover/selection details (new Reqs)
- Significantly expanded/enhanced text for proxies (new recommendations)
- BRSKI-PLEDGE enhancements text (new detail requirements)
- DNS-SD, CORE-LF encoding detail enhancements (new detail requirements)
- IANA requests refinemenets
- Bug fixes

Discovering of BRSKI pledges

Requirements

- BRSKI-PRM requires pledges to be discoverable by a name derived from their IDevID.
- Unless not desired by pledge:
  “Browse” for pledges even if IDevID is not known.

Solution

- Vendor needs to publish scheme how to create <Instance> field of service instance name
  - Make it easy to start using BRSKI even if manufacturer IDevID certs have information required in different places than we would like them to be in
- Recommendation to compose from IDevID X520SerialNumber and domain name “owned” by manufacturer
  - Simple serial numbers coul otherwise clash across manufacturers.
- Explicit/expanded restatement of RFC8995 assumed requirement
  - Recommendation to compose X520SerialNumber so that it can be composed from non-cryptographic known device identification (from sales receipt or the like)
- Ask for support of mDNS and (MAY) DNS-SD SRP
  - Q to DNS-SD group: Does DNS-SD SRP have enough autoconfig so it will work well enough on “pledges” (devices) with only DHCP learned config...

Example

Pledge IDevID certificate information:

```
; Format as shown by e.g.: openssh
  Subject: serialNumber = "PID:Model-0815 SN:WLDPC2117A99",
    O = example.com, CN = Model-0815
```

Manufacturer published LDevID identification schema: (New in -05)

```
 Brand: Example: O = example.com, serialNumber = "PID:<PID> SN:<SN>"
 SN = Serial Number, PID = Product Identifier
```

Manufacturer published DNS-SD Instance Schema:

```
 <X520SerialNumer>.<O>
```

DNS-SD Instance:

```
  "PID:Model-0815 SN:WLDPC2117A99.example.com"
```

DNS-SD RR for the pledge (using mDNS, hence .local):

```
  ; PTR RR to support browsing / discovery of service instance name
  _brski-pledge._tcp.local  IN PTR
    PID:Model-0815\\ SN:WLDPC2117A99\\.example\\.com._brski-pledge._tcp.local
   …
```

BRSKI PLEDGE open questions (1)

- Not (yet) defined how to do BRSKI Pledge discovery via GRASP or CORE-LF
  - Do we want to add this ?

- Missing feature: concept of (browsable) service instance name in GRASP, CORE-LF
- GRASP (ad-hoc):
  - Could re-use Objective Name (service name) as service instance name:
  - Objective for finding specific pledge: IDevID.<Instance>
  - Objective for browsing: IDevID
  - Objective Value encodes service instance name and variations supported.
- GRASP (generic):
  - Revive *draft-eckert-anima-grasp-dnssd*
  - Preferreed (by author)
- CORE-LF
  - Generic mapping of DNS-SD into CORE-LF beyond scope of useful work now ?
  - One-off mapping needed ?

BRSKI PLEDGE open questions (2)

- Current text suggests but does not formally specify:
  - Scheme for secure assignment of service instance names
  - Added warning about problem intos ecurity section
- Manufacturer root-CA (or sub-CA thereof)
  - Needs a WebPKI signature attesting to the following domain name in the Cert
    “*.idevid.<mydomain>”
- Pledge IDevID signed by this manufacturer Cert
- This implies the service instance name of the pledge is
  “<X520SerialNumber>.idevid.<mydomain>”

- Web root of trust will give “*.idevid.<mydomain>” only to one manufacturer
  - Hence this proves “authoritatively” that pledge “owns” the service instance name
  - *A DNS-SD SRP server could authenticate this – and reject registration of names from fake devices!*

- Using the explicit “.idevid.” domain name component may be an ugly hack.. (or not ?)
  - There are probably other encoding options to skin this cat.

Opinions ?

Redundant discovery and selection

New section

Discovery and Selection by initiators

- Newly written (coalesced) to create more precise requirements / recommendation
  - Primarily for pledges/proxies discovering/selecting Registrar/Proxies
  - Equally applicable for any future selectable servers in BRSKI (or beyond)
  - Could not locate any good RFC to quote. Timeouts etc somewhat application dependent (DNS-SD / 6MAN RFC…s)
  - DNS-SD originates from human selection/browsing use-cases. Automated use DNS-SD for service redundancy far less well explored from operatinal details ?!
- Requirement to be able to sequentially connect to multiple discovered responders
- Wait 30 seconds after failing to connect to any discovered responders before restarting
- Support for ordering discovered responders by priority and weight – and filtering by suitable variation
  - Introduce term “preference” to allow ordering by more versus less preferred variations
- Pick-at-random after reboot between equal preference responders (to avoid recurring failures if one responder causes problems which will make pledge fail to continue
  - Most likely cause of many responders railing is bad network breakage.
- Limits for how many responders must be supportable: At least 4, but should not attempt to connect to more than 10 top discovered ones
- When connection times out to a responder, remove it from candidate list for 120 seconds
- Recommendation against responders to remove their service announcements when they are not responsive
  - Can be significant work to implement though, need probing agent on server itself to see if serivce process is not responsive.

Special considerations for proxies

- Stateful proxies are initators just like pledges
  - Except they will create more connections
  - So they should remember responder reachability across connecting pledges

- Stateless proxies do not have explicit protocol to learn responder unresponsiveness
  - So they must observe _ANY_ packets returned by selected responder
  - And if no packets are seens in response (30 seconds window) select another (next-best) responder instead.
- Using newly discovered better resoonder in stateless proxy must be very careful
  - Do not do this in the middle of a pledge talking to the responder
  - Recommendation to only add such a new responder after 120 seconds total idling
  - Helps for any aliveness messages from pledges (please wait for voucher...): safe if refrehs are sent once per minute

Variation suport in proxies

Summary

- Rewritten / clarified / more implementer help
- Main goal
  - automatic creation of service announcements for any of the variations (known or future)
  - Correct proxying of pledge to a registrar that does support the desired variation
- Propose / discuss different implementation models
  - Direct Connection Mode
    - Create new proxy responder socket for every discovered registrar socket and announce same variations (but with proxy service name)
    - Limit total number of offered sockets to e.g.: 10 best discovered registrar sockets (per non-overlapping variation set)
  - Best Registrar Selection Mode
    - Only create single socket for every unique set of variations that require selection of different registrars
    - Dynamically select best responding registrar for the socket, also based on (un)responsiveness
  - Proxy in name only on registrars
    - Only announce registrar as proxy, no actual forwarding plane of their own
    - Discusses limitations (not sufficient as generic proxies on the edge, e.g.: If this registrars MASA connectin fails)
    - Impediment to introduction of new variations (from other registrars) if deployed on edge (only deploy in core).
    - May still want to dynamically self-probe registrar to dynamically announce/withdraw service announcement
- Scalability discussions
  - Registrar/Proxy in every site (of large enterprise) – for high availability
    - IMHO Open issue for every discov ery method
    - Requires operational hacks (GRASP filtering, per-site DNS domain prefixes or .local per-site, …)

DNS-SD (new) details

Refined DNS-SD details / guidance

- Variation signaling:
  - Every variation encoded as DNS-SD TXT “key” with binary value. Using optimized “key” to indicate “key=1”. TXT therefore has list of variations as value. May have other, non-variation keys of course.
- Requirements to automatically create service instance name
  - Discuss choices, e.g.: MAC-address-string and when it is really uniqe
- Requirement to automatically create host name
  - But prefer/use pre-existing host name if it exists
- Derive service instance name from such existing host-name by appending name components (e.g.: “-registrar”).
  - Add “-<pid>” if service is known to be process of which there may be multiple instances running
- Several more hopefully helpful recommendations

Refined DNS-SD details / guidance

- Variation signaling:
  - Every variation encoded as DNS-SD TXT “key” with binary value. Using optimized “key” to indicate “key=1”. TXT therefore has list of variations as value. May have other, non-variation keys of course.
- Requirements to automatically create service instance name
  - Discuss choices, e.g.: MAC-address-string and when it is really uniqe
- Requirement to automatically create host name
  - But prefer/use pre-existing host name if it exists
  - Likely: proxies will not have them, registrars will.
- Derive service instance name from such existing host-name by appending name components (e.g.: “-registrar”).
  - Add “-<pid>” if service is known to be process of which there may be multiple instances running
- Several more hopefully helpful recommendations

Examples

```
# BRSKI - automatically generated instsance/host names - mDNS (.local)
_brski-registrar._tcp.local
                    IN PTR  0000-5e00-5314._brski-registrar._tcp.local
  0000-5e00-5314._brski-registrar._tcp.local
                    IN SRV  1 2 4555 0000-5e00-5314.local
  0000-5e00-5314._brski-registrar._tcp.local
                    IN TXT  "est-tls" "prm-jose" "cmp"


# cBRSKI
   _brski-registrar._udp.local
                     IN PTR  0000-5e00-5314._brski-registrar._udp.local
   0000-5e00-5314._brski-registrar._udp.local
                     IN SRV  1 2 5684 0000-5e00-5314.local
   0000-5e00-5314._brski-registrar._udp.local
                     IN TXT  "rrm-cose"


# Host name - automatically generated by “OS”
  0000-5e00-5314.local
                    IN AAAA  2001:DB8:0815::5e00:5314
```

Examples

```
# BRSKI registrar - applications - pre-known hostname - SRP / default-domain


_brski-registrar._tcp.example.org
     IN PTR  noc-registrar-brski-37253._brski-registrar._tcp.example.org
noc-registrar-brski-37253._brski-registrar._tcp.example.org
     IN SRV  1 2 4555 noc-registrar.example.org
noc-registrar-brski-37253._brski-registrar._tcp.example.org
     IN TXT "est-tls" "cmp"


_brski-registrar._tcp.example.org
     IN PTR noc-registrar-prm-9735._brski-registrar._tcp.example.org
noc-registrar-prm-9735._brski-registrar._tcp.example.org
     IN SRV 1 2 17355 noc-registrar.example.org
noc-registrar-prm-9735._brski-registrar._tcp.example.org
     IN TXT "prm"


# Host name
noc-registrar.example.org
     IN AAAA  2001:DB8:0815::5e00:5333
```

CORE-LF (new) details

Refined CORE-LF encoding

REQ: GET /.well-known/core?rt=brski.*

RES: 2.05 Content
<scheme://[address]:port path-abempty>;\
rt=brski-service(;var="brski-variation-string(s)");\
pw="priority weight"

TBD: Not sure if this formatting is best way to specify format, but full ABNF constraints are ugly.

Mandate use of IP/IPv6 addresses (avoid need to support DNS in pledges)

- Mandate inclusion of address even if not necessary to make data easier portable (outside context of connection).

Added new attribute “pw” to encode priority and weight for selection, just like in DNS-SD

Recommend all rt (resource targets) start with “brski.”

- But currently no known “reservations” for such prefix in IANA registry. Draft is asking for one. Let’s see...

Variations previously used “bv” attribute. Now changed to “var”.

- To be registered via IANA ask anyhow. If other protocols/services than brski want to use same attribute they are welcome

Variations encoding: space (“ “) concatenated into single quoted string.

CORE-LF example

```
REQ: GET /.well-known/core?rt=brski.*

RES: 2.05 Content
Content-Format: 40
Payload:
 <https://[2001:DB8:0815::5e00:5314]:4555>;          # [1]
        rt=brski.rs;var="est-tls prm-jose cmp";
        pw="1 2",
 <https://[2001:DB8:0815::5e00:5314]:4555>;          # [2]
        rt=brski.jp;var="est-tls prm-jose cmp";
        pw="1 2",
 <coaps://[2001:DB8:0815::5e00:5314]:5684/b>;        # [3]
        rt=brski.rs;var=,
        pw="1 2",
 <coaps://[2001:DB8:0815::5e00:5314]:5684/b>;        # [4]
        rt=brski.jp;var=,
        pw="1 2",
 <coaps+jpy://[2001:DB8:0815::5e00:5314]:6534/b>;    # [5]
        rt=brski.rjpy;var=,
        pw="1 2"
```

Questions ?

- Thank you!