

Network Working Group
INTERNET-DRAFT
Updates: 7585
Category: Standards Track
Expires: September 05, 2022

DeKok, Alan
Network RADIUS
5 March 2022

EAP Usability
draft-dekok-emu-eap-usability-01.txt

Abstract

This document defines methods which enable simpler deployment of TLS-based EAP methods. It defines new certificate fields, and uses existing certificate fields in order describe new methods for bootstrapping security. The methods defined here change TLS-based EAP supplicant configuration from a complex and insecure process to one that is automated, and is essentially trivial. These methods are still, however, compatible with existing standards and practices.

Status of this Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at <http://www.ietf.org/ietf/lid-abstracts.txt>.

The list of Internet-Draft Shadow Directories can be accessed at <http://www.ietf.org/shadow.html>.

This Internet-Draft will expire on January 12, 2022.

Copyright Notice

Copyright (c) 2022 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 (<http://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	5
1.1. Requirements Language	6
2. Historical Problems	7
2.1. Supplicant Changes over Time	7
2.1.1. Phone Vendor One	7
2.1.2. Phone Vendor Two	8
2.1.3. Operating System Vendor One	9
2.1.4. Operating System Vendor Two	9
2.1.5. Operating System Vendor Three	9
2.2. Problems with Certificates	10
2.2.1. Problematic Use of Certificate Stores	10
2.2.2. Problematic use of key purpose fields	11
2.2.3. Problematic Use of Certificates	12
2.2.4. Obtaining Certificates with the new OIDs	13
3. Principles and Guidelines	15
3.1. Network Configuration Guidelines	15
4. New Recommendations for Certificates with EAP	17
4.1. Comparison to HTTPS	17
4.2. Additional Information Required	18
5. How It Works in Practice	19
5.1. A worked example for EAP-TLS	19
5.1.1. The Problem Statement	19
5.1.2. Obtaining the Certificates	20
5.1.3. Configuring the end user device	21
5.1.4. Obtaining Network Access	22
5.2. Other TLS-based EAP methods	23
5.3. EAP methods which do not use TLS	24
5.4. Other Methods of Provisioning	24
5.5. Trust on First Use Can be Secure	25
5.6. Additional Considerations	26
6. Extending the Solution	28
6.1. Bootstrapping via EST	28
6.1.1. Closing the loop	29
6.2. Bootstrapping via DNS	30
6.2.1. CERT records	30
6.2.2. CERT record labels for Server Certificates	32
6.2.3. CERT record labels for CA Certificates	32
7. Related issues	33
7.1. Provisioning Issues	33
7.1.1. Bootstrapping via a Separate Network	33
7.1.2. Configuration Change is just Refresh	34
7.1.3. Secure versus Insecure Provisioning	35
7.2. Issues related to Security	36
7.2.1. Why id-on-naiRealm	36
7.2.2. Resumption	36
7.2.3. Choosing EAP Types	37

7.2.4. User Experience	37
7.3. Issues related to Certificates	38
7.3.1. Public CA versus Private CA	38
7.3.2. Limitations of public CAs	39
7.3.3. CA Chains	40
7.3.4. Delegated Authentication	41
7.3.5. Identification of Networks	42
7.4. Anti-solutions	42
7.4.1. MDM Products Are not the Solution	42
7.4.2. EST and similar protocols do not solve all of th	44
7.4.3. Captive Portals and Hotspots	45
7.4.4. Fully Anonymous Network Access and Provisioning	45
7.5. id-kp-eapOverLAN May not be sufficient	46
7.6. Guest Networks	46
7.7. Using TLS with protocols other than EAP	48
8. Moving to the new methods	49
8.1. Using the new OIDs	49
8.2. Recommendations for EAP peers and authenticators	50
8.3. Principles and Guidelines	52
9. Security Considerations	53
9.1. On Identities and Service Discovery	53
9.2. Password Hashing and Storage	53
10. IANA Considerations	54
10.1. Key Purpose OIDs	55
10.2. Underscored and Globally Scoped DNS Node Names	55
11. References	55
11.1. Normative References	55
11.2. Informative References	56

1. Introduction

TLS [RFC8446] has been widely deployed, and is used with EAP [RFC3748] and with RADIUS [RFC2865]. Historically, these specifications have been written to define the protocols "on the wire", with minimal description of use-cases and usability. The success of these specifications has been that perhaps a billion devices use EAP. The failure of these specifications is that EAP can be still difficult to use, both for administrators and for end users.

Even with a clear standard, implementations do not always follow the specifications exactly. In some cases implementations do less than what is recommended, which can cause security and inter-operability issues. In other cases, implementors do more than what is recommended, as they have found the specifications insufficient to address practical requirements. In other cases, there is no standard, so implementors make individual choices as to how their implementations work. Even worse, implementors change their implementations over time, to solve problems which are not addressed in the standards.

All of these issues lead to confusion for end users and administrators. These issues lead to decreased security of the protocols, and decreased trust in the protocols.

The result of the above problems is software where many critical aspects of its operation are vendor-defined. This wide variation gives a poor experience for all parties involved, and contributes to decreased security.

This document therefore defines method to enable the simple and easy deployment of TLS-based EAP methods. It defines new certificate fields, and describes how those fields can be used to gain network access easily and securely. The processes it defines are clear and straightforward. The end user experience is understandable, and difficult to get wrong.

That is, this specification removes the need to rely on end users to make security decisions. History shows us that such reliance is misplaced. Instead, we rely on the global certificate system which has proven to work well, along with a few changes to the behavior of EAP systems.

These ideas are not new. [RFC4334] Section 1 says:

Automated selection of client certificates for use with PPP and IEEE 802.1X is highly desirable. By using certificate extensions to identify the intended environment for a particular certificate,

the need for user input is minimized.

We extend the above statements to include server certificates, and to further define automated processes by which TLS-based EAP methods can be configured. In addition, where [RFC4334] describes the "automated selection of client certificates", we invert that use-case to show how certificates can be used to automate network configuration, via a set of simple and clearly defined processes.

The requirements on the client device are simple:

- * it has some kind of network connectivity,
- * it has a root certificates for the web,
- * it knows the domain for which it wishes to authenticate

e.g. "example.com", * it knows a username and password at that domain.

With that information, the configuration process can be automated, with essentially zero user input.

The document begins with an overview of the current situation. We describe the problems which have motivated this document. First by showing how the behavior of multiple vendor implementations has changed over time. We then discuss problems with the ways that certificate are handled. We discuss how the standards contradict each other, and how current practices contradict, ignore, or extend the standards in incompatible ways.

The document begins with a worked example, initially just for EAP-TLS, and then showing how the processes described for EAP-TLS can be easily applied to other EAP methods.

We extend the given solution by using DNS and HTTPS to perform initial bootstrapping of the supplicant configuration. We show how supplicants can be configured securely by leveraging the existing trust in the web PKI. This bootstrapping requires no changes to supplicant code or behavior.

We finally summarize the work by giving a set of specific recommendations.

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. Historical Problems

There have historically been a number of issues with configuring devices for EAP authentication. The overly positive description of this history is that it has resulted in a wide variety of tools and products available to configure EAP on end-user devices. In addition to the wide variety of configuration products, the behavior of native supplicants has also varied widely over time.

These issues point to an underlying problem which has, as yet, been unresolved. Each vendor of configuration products or devices to be configured has largely been performing searches by "trial and error" in order to find the best user experience. The result, of course, has been a frustrating experience for both users and for vendors.

We do not discuss Mobile Device Management (MDM) vendors or vendors of other supplicant configuration products here. Their products are largely tools which use the APIs presented by supplicant vendors, and attempt to hide the complexity from end users.

We also do not discuss the behavior of EAP servers (authenticators) or RADIUS servers. The issues seen by supplicants are largely related to user experience, and have little effect on the "on the wire" protocol. As such, RADIUS servers have been required to make correspondingly fewer changes to their implementations. In addition, RADIUS servers are generally designed to be complex, with complex policies. These policies enable their administrators to change the behavior of the software without resorting to product upgrades.

As a result, we focus initially on supplicants, how their behavior has changed over time, and the publicly visible effects of those changes.

2.1. Supplicant Changes over Time

In this section we discuss how the behavior of multiple supplicants has changed over time. We do not name the vendors of these supplicants, as there is no need to blame them for being unable to solve an industry-wide problem.

2.1.1. Phone Vendor One

This vendor has been gradually disabling the supplicant configuration API. Instead, configuration product vendors are able to influence the user interface with suggested prompts at different points in the authentication framework.

It appears that the goal is to give the user control over the

network. A related goal is to require the user's consent before making changes to the network.

The system also treats SSIDs as defining a location, even though SSIDs are not inherently location-specific. This mislabelling means that users' are shown prompts about location, when in fact the operation being performed is connecting to an SSID.

The effect of these issues is that the user is unable to meaningfully consent. There may insufficient information available, the available information may be meaningless to the end user, or the information given to the end user is simply wrong.

This vendor has changed how manual connections are managed over time. The user is not always prompted, but the systems behavior has gone through the following changes to connection requirements:

- * do not perform certificate validation
- * validate that the root CA is in the system certificate store
- * validate that the root CA is in the WiFi certificate store
- * require a DNS name for the RADIUS server.

None of these solutions are optimal.

2.1.2. Phone Vendor Two

This vendor has a standard format for WiFi configuration files. The user can manually install the configuration, but that configuration is not active until an additional manual step is performed to enable it.

A standard configuration is useful, but the configuration file is not typically signed, even though that is supported. It appears that the the manual enablement step is an attempt to work around the lack of authentication for the configuration files.

As was seen in the previous section, the end user has no meaningful information about the configuration. This lack of information means that the user is conditioned to simply accept the configuration and enable it, without paying attention to its contents.

This vendor does, however, provide a robust API. This API permits a rich ecosystem of MDM products which automate the configuration of end-user devices.

2.1.3. Operating System Vendor One

This vendor has been gradually disabling their WLAN API. The remaining APIs permit MDM solutions to associate a user identity with a particular network. However, there is no ability to set a root CA or a RADIUS server DNS host name.

When the user connects to the network, a prompt is shown which asks the user if the server is valid. The server certificate is presented to the user, but the user has no information about which root CA is acceptable or not. Instead, the user is shown fields from an unknown certificate, and is asked to validate that the certificate is acceptable.

Again, the user has insufficient information to meaningfully consent. Which again means that the only course of action is to mindlessly click on "accept".

2.1.4. Operating System Vendor Two

This vendor retains a rich WLAN API, but has removed the ability to configure specific users. Instead, only system-wide profiles can be set.

This vendor provides for easy installation of additional root CAs, but those root CAs are permitted to be used for any purpose. Which means that a malicious private CA can issue HTTPS certificates for any domain. These fake certificates will be silently accepted by the systems browser as being valid. The user will be unable to distinguish malicious sites presenting those fake certificates from the genuine domains.

It is difficult to overstate the negative security impact of that process.

An MDM product adding a private CA generally requires a privileged account to install the CA. A user can install a CA manually, but the operating system will show the user large amounts of text in order to warn the user about the security issues of this process. The user, of course, has no way of understanding many of these warnings, and is left again to mindlessly click on "accept".

2.1.5. Operating System Vendor Three

This vendor retains a rich WLAN API, and a number of tools by which network configuration can be performed. These tools are widely used by MDM vendors to automate the configuration of networks.

However, the end user experience is still complex. The user still must manually select each individual parameter from multiple options. This capability gives the user a substantial amount of control over the process, but does not provide any more information than is available in other operating systems.

This process of allowing the end user to configure everything is useful for experienced and knowledgeable users. However, it leaves the average user with a bewildering set of choices, most of which are meaningless or opaque. The user is then left to mindlessly follow online guides, which may or may not work, and which do not give the user enough information to give informed consent for the actions that are being taken.

2.2. Problems with Certificates

The widely (and wildly) changing behavior of supplicant software is not the end of the story, however. There are a large number of problems related to the use and abuse of certificates. These issues are discussed in more detail in this section.

2.2.1. Problematic Use of Certificate Stores

Some EAP peers use a different certificate store for EAP than for other (e.g. web) applications. In practice, the use-case of "downloading video from a known source" is substantially different from the use-case of "sending authentication credentials to a known destination". As such, the certificate stores should be different for these two use-cases.

When a CA is allowed to be used for EAP, then the implication is that all certificates signed by that CA are allowed to be used for EAP. This result is not secure. It permits attackers to get a valid server certificate from a public CA, and then to set up an EAP server. Naive EAP peers will then send user credentials to the malicious server. Worse, there is no general way for any third-party to detect that this impersonation has happened. It is only visible to EAP peers who are in a small geographic area.

Tests have shown that in a university environment, up to fifty percent of EAP peers will connect to a malicious SSID without checking the CA or server certificate. In effect, these peers will send authentication credentials to anyone who asks.

The security problems associated with such behavior cannot be overstated.

However, not all EAP peers uses separate certificate stores. Using one certificate store is less of an issue when "self signed" or

"private" CAs are used. The use of private CAs for EAP means that the EAP system is now more secure. However, the addition of private CAs to a global certificate store means that those private CAs can now issue certificates for well-known public web sites. The possibility of such forgery has made it difficult for MDM vendors or site administrators to create and use private CAs.

As such, we reiterate that the certificate stores SHOULD be different for these each application or use-case. Where the system cannot tolerate multiple different stores, it SHOULD at least mark each CA certificate with an annotation as to its intended purpose. While there is a "key purpose" field defined for certificates, we will see that this field is not always suitable for differentiating certificate purposes.

2.2.2. Problematic use of key purpose fields

[RFC5216] Section 5.3 makes the following recommendations about the certificate used by the EAP-TLS server:

In the case of the EAP-TLS peer, this involves ensuring that the certificate presented by the EAP-TLS server was intended to be used as a server certificate. Implementations SHOULD use the Extended Key Usage (see Section 4.2.1.13 of RFC3280) extension and ensure that at least one of the following is true:

- 1) The certificate issuer included no Extended Key Usage identifiers in the certificate.
- 2) The issuer included the anyExtendedKeyUsage identifier in the certificate ...
- 3) The issuer included the id-kp-serverAuth identifier in the certificate ...

These recommendations have also been used in EAP-TLS [EAPTLS], EAP-TTLS [RFC5281], PEAP [PEAP] and [MSPEAP], EAP-FAST [RFC4851], and TEAP [RFC7170].

We first note that this document extends and strengthens the suggestion that systems ensure "that the certificate presented by the EAP-TLS server was intended to be used as a server certificate." We also extend this recommendation to client certificates, further strengthening the security around TLS-based EAP methods.

However there is an issue with the [RFC5216] recommendations. These recommendations appear to be in direct conflict with the definition of id-kp-serverAuth from [RFC5280] 4.2.1.12, and of the requirements for its usage:

If the extension is present, then the certificate MUST only be

used for one of the purposes indicated. If multiple purposes are indicated the application need not recognize all purposes indicated, as long as the intended purpose is present.

... If a certificate contains both a key usage extension and an extended key usage extension, then both extensions MUST be processed independently and the certificate MUST only be used for a purpose consistent with both extensions. If there is no purpose consistent with both extensions, then the certificate MUST NOT be used for any purpose.

```
...
id-kp-serverAuth          OBJECT IDENTIFIER ::= { id-kp 1 }
-- TLS WWW server authentication
```

This definition shows that id-kp-serverAuth is intended for "WWW server" usage, which is in conflict with how it is defined in [RFC5216]. Similar issues exist for the id-kp-clientAuth OID, in that it is intended for "WWW client", which is not correct for EAP.

It appears that the recommendations made in [RFC5216] were taken from [RFC2716], and were made for entirely practical reasons. The desire was for users of EAP to be able to obtain certificates from public root CAs. However, those root CAs could not (or would not) issue certificates which contained OIDs other than id-kp-serverAuth. Therefore as work around, [RFC5216] Section 5.3 allowed for a wide variety of EKUs to be used in server certificates. These certificates could then come from private CAs, or from publicly known root CAs.

We believe that the long-term correct solution is to define and use additional key purpose OIDs. These key purpose OIDs can initially be used by EAP implementations along with private CAs. As support for these OIDs becomes more widely available, it may be possible for public CAs to issue purpose-specific certificates.

The problematic use of id-kp-serverAuth has had a number of impacts, past the issue of contradictory specifications. These impacts result in certificates being used in problematic ways, which we discuss below.

2.2.3. Problematic Use of Certificates

The current workarounds to the contradictions in the specifications are two-fold. One, is simply to get certificates with id-kp-serverAuth from a public CA, and hope that using it for EAP either is acceptable, or that it is not noticed. Another is to use a self-signed CA. Both work-arounds have problems.

Many people prefer to use public CAs, as they are seen as "better"

than self-signed CAs. However, using a public CA likely means violating the terms of use of that CA. Which means that the network continues to work so long as this mis-use is not reported.

It can be useful instead to use private CA. A private CA can add id-kp-serverAuth without violating any terms of use, or it can omit the key purpose OIDs, or it can add custom key purpose OIDs.

However, in addition to the problems noted in the earlier section, private CAs are not installed by default in client devices. This limitation means that these CAs must be provisioned somehow. As seen above, these provisioning methods can be complex and prone to failure.

As such, there is no simple, easy, way for administrators to both obtain and provision certificates for use with EAP.

We also note that these OIDs are used not only for EAP, but that they are also used for other public-facing TLS services such as XMPP, SMTPS, LDAPS, etc. Those protocols may have similar issues with alleged mis-use of these OIDs. If these use-cases are forbidden user CAB guidelines, then this would seem to be a serious problem with the global certificate framework.

We leave the solution of these issues as a point of discussion for the wider Internet community.

2.2.4. Obtaining Certificates with the new OIDs

Most CAs currently offer limited support for non-"WWW" OIDs in certificates. In many cases, the Certificate Signing Request (CSR) supplied by the customer is (in practice) used only as a vague suggestion. While the CA generally does not add any fields, it may drop fields that it does not recognize or support. Or, the CA may discard the CSR entirely.

[CAB] Section 7.1.2.3 makes it difficult for existing CAs to issue client certificates which contain the new OIDs:

Either the value id-kp-serverAuth [RFC5280] or id-kp-clientAuth [RFC5280] or both values MUST be present. id-kp-emailProtection [RFC5280] MAY be present. Other values SHOULD NOT be present. The value anyExtendedKeyUsage MUST NOT be present.

Further, the requirements of [CAB] Section 7.1.2.4 essentially forbids CAs from signing certificates which are intended for use with EAP:

CAs SHALL NOT issue a Certificate with:

- a. Extensions that do not apply in the context of the public Internet (such as an extKeyUsage value for a service that is only valid in the context of a privately managed network), unless:

None of the reasons listed after "unless" allow for CAs to issue certificates for use with EAP in a privately managed network.

This behavior by CAs makes it difficult in practice, if not impossible, to obtain non-"WWW" certificates from a public CA.

The suggestion given here is to simply use self-signed CAs. This suggestion is not always practical.

It is possible to define CAs for "walled gardens" with a private CA. One example is certificates used internally in an organization, or in a group such as Eduroam [RFC7593] and [EDUROAM]. In those situations, the members requesting certificates have already validated, and there is already a legal framework in place to protect the parties.

Other suggestions have been that it is relatively simple to set up a new CA, with new procedures and requirements. Given the regulatory requirements around CAs, it appears that new public-facing CAs have to be well funded. i.e. requiring many millions of dollars. It is therefore difficult, if not impossible, for small public-facing CAs to be created.

The goal of this document is to permit better behavior for EAP peers and authenticators. If this specification is widely deployed, then there may be sufficient demand for CAs to offer new certificates which are marked as fit for their intended purpose.

3. Principles and Guidelines

After analysis of the historical practices and standards for EAP, we came to a set of guidelines which are outlined in this section. Application of these guidelines drove the rest of the specification which we define herein.

3.1. Network Configuration Guidelines

It is RECOMMENDED that the guidelines given below are followed when developing new network configuration standards and methods:

- * Automated provisioning is strongly preferred to manual provisioning. We define "automated provisioning" as provisioning which is performed via software, with little or no user intervention. i.e. "zero touch" for end users. Automation minimizes the possibility for end users to create broken or insecure configurations.
- * Manual provisioning should be limited to "Trust on first use" (ToFU), and cached or "pinned" after that. That is, manual provisioning should be limited to allowing a user to approve validation decisions which have been made by the system.
- * Relying on end users to manually configure complex systems is strongly discouraged. Complex systems are difficult to configure, and improperly configured systems create many issues related to security, usability, and network access.
- * Configuration should be "pinned" in order to permit systems to detect and prevent unauthorized changes, and to detect malicious networks which claim to be updated versions of the true network.
- * The identity and role of both parties should be exchanged, and verified. In practice, this suggestion often means that TLS-based EAP methods are preferred to ones which only do name / password credential verification. TLS allows both parties to exchange certificates which demonstrate their identities,
- * The previous requirement usually means that the both parties know which RFC 7542 NAI realm is being used. This realm serves a similar purpose to the the DNS host name used in other TLS-based protocols such as HTTPS. As such, similar methods can be used to validate certificate authenticity. This NAI realm is contained in an id-on-naiRealm field, as defined in [RFC7585] Section 2.2
- * For TLS-based EAP methods, trust should be based on a certification authority (CA), which signs certificates for a

particular realm. If the CA is trusted, then everything derived from that CA can be trusted. If the CA is not trusted, then it is impossible to trust anything derived from an untrusted CA.

- * CAs should also be associated with permitted uses. For example, a root CA which is trusted for web surfing is not necessarily trusted for use with EAP authentication. In practice this means either having separate certificate stores for different purposes, or annotating root certificates with their permitted uses. Again, this process should be automated, because end users have no way to verify which CA is valid for what purpose.

We believe that these recommendations are correct, simple, practical, and will improve security and usability for all participants in EAP. We show that there is a clear upgrade path from current behavior to better behavior. Each step of that upgrade path is simple, and involves minimal change for end users or administrators.

4. New Recommendations for Certificates with EAP

The first step towards a complete solution is to define new OIDs. These OIDs indicate that certificates are intended for use with an EAP server, or an EAP peer.

The following key usage purposes are defined within id-kp:

```
id-kp OBJECT IDENTIFIER ::= { id-pkix 3 }
```

```
id-kp-eapServer          OBJECT IDENTIFIER ::= { id-kp TBD-1 }  
-- TLS EAP server authentication  
-- Key usage bits that may be consistent: digitalSignature,  
-- keyEncipherment or keyAgreement
```

```
id-kp-eapClient          OBJECT IDENTIFIER ::= { id-kp TBD-2 }  
-- TLS EAP peer authentication  
-- Key usage bits that may be consistent: digitalSignature,  
-- and/or keyAgreement
```

These EKU fields mirror id-kp-serverAuth, and id-kp-clientAuth, respectively.

We also rely on id-on-naiRealm, as defined in [RFC7585] Section 2.2. This field contains the NAI realm [RFC7542] in which the user has an identity, and for which the EAP server is performing authentication.

4.1. Comparison to HTTPS

We can further explain how these fields help EAP by comparison with how certificates are used for HTTPS [RFC2818]:

- * HTTPS uses id-kp-serverAuth to indicate that a certificate is permitted to be used with an HTTPS server. We define id-kp-eapServer which indicates that a certificate is permitted to be used with an EAP server.
- * HTTPS uses id-kp-clientAuth to indicate that a certificate is permitted to be used with an HTTPS client. We define id-kp-eapClient which indicates that a certificate is permitted to be used with an EAP client.
- * HTTPS uses id-ce-subjectAltName with dNSName [RFC5280] to contain the DNS name of the server to which the client is connecting. We use id-on-naiRealm [RFC7585] to indicate the NAI realm of the server to which the

client is authenticating.

4.2. Additional Information Required

When combined with a clearly defined process, the above definitions allow devices to use TLS-based EAP methods with no more complexity than is seen when browsing the web. That is, in many situations, all the end device needs is the following:

* network access (trusted or not) * an account in a domain, e.g.
"user@example.com" * one or more trusted root CAs from the web PKI.

This information can be used to securely obtain network access. The procedures outlined here work with both public CAs and private CAs.

We will first describe how these fields can be used to make EAP authentication easier to use. Once we have described a worked example using these fields, we will show how to extend the solution to solve the remaining open issues.

5. How It Works in Practice

In this section we provide a worked example for EAP-TLS. This discussion uses EAP-TLS as an example, but the methods discussed here are not limited to that use-case. Describing a specific EAP method allows us to discuss every aspect of the proposal, without worrying about how similar methods are applicable to different situations.

We expand the discussion later in this section to show how these methods are applicable to other TLS-based EAP methods.

5.1. A worked example for EAP-TLS

We explain how this specification works via an example using EAP-TLS. We start off with the problem statement, then how the certificates might be obtained, then how the EAP peer is configured using information in the certificates, and finally how the device obtains network access.

5.1.1. The Problem Statement

For the initial worked example, we assume that we are trying to solve the limited problem of an end user who has a WiFi enabled device such as a laptop. The user wishes to use that device to get online, via the simplest possible method. There is an administrator who also wishes to get that user online, and wishes to configure the end user device to do EAP-TLS.

We also presume that there are some additional pieces of the solution, as follows:

- * There may be a Wireless LAN (WLAN) System Service identifier (SSID) which is used for WiFi authentication. This information can be realized in the certificate field `id-pe-wlanSSID`, as defined in [RFC4334] Section 3.
- * All parties know which NAI realm [RFC7542] is being being used. For example, an individual may work for a company "example.com". This information is placed into the certificate field `id-on-naiRealm`, as defined in [RFC7585] Section 2.2.
- * We use the new EKU field `id-kp-eapServer`. This field indicates that a certificate is intended to be used as a server certificate within EAP.
- * We use the new EKU field `id-kp-eapClient`. This field indicates that a certificate is intended to be used as a client certificate within EAP.

Knowing the name of the SSID is necessary, but perhaps not sufficient. Some environments may have additional security requirements, such as mandating that only WPA3-192-bit connections may be used. This information should likely go into another field of the certificate. For the purposes of this document, we will assume that knowing the SSID name is acceptable.

EAP methods are also used to authenticate users when SSIDs are not available (e.g. wired 802.1X), the use of `id-pe-wlanSSID` is recommended, but is not required. For the purpose of this section, we assume that WiFi access is being configured. Later discussion will show how the methods outlined here can be applied to other forms of network access.

As we will see below, this information is sufficient to configure EAP-TLS securely, and with minimal effort by the end user.

5.1.2. Obtaining the Certificates

The administrator begins by obtaining a server certificate from a root CA. This CA can be public or private. The only requirement is that the CA is willing to sign certificates which contain an `id-on-naiRealm` field, and which also contain the `id-kp-eapServer` field which indicates that this certificate is suitable for use with EAP. The rest of the fields, and the validation process for those fields, can be identical to the processes used today.

The use of the new EKU fields here is intended to illustrate the end goal of simplifying deployments. As we will see later, there are intermediate steps which do not require the new EKU fields.

The administrator also obtains a client certificate, which will be given to the end user for installation on the device. This client certificate is issued via a hierarchy which ends in a known root CA. The rest of the certificate hierarchy here is not important to this example. We only presume that it exists; that it is valid; and therefore that it is trusted.

The client certificate contains an `id-on-naiRealm` field, which has the same NAI realm as seen in the server certificate. The client certificate also contains the `id-kp-eapClient` field which indicates that this certificate is suitable for use with EAP. The client certificate may also contain a `id-pe-wlanSSID` field, though this is not required.

We presume that the client certificate also has an associated private key, and that this key is encrypted using a password.

We now have enough information to configure the end user device.

5.1.3. Configuring the end user device

The administrator configures the end device with the client certificate, along with its associated private key and password. For the purposes of this example, it is not important how the device obtains that information. As noted earlier, the distribution problem will be solved later in this document by extending the solution.

The configuration may also include the entire certificate chain up to, and including, the root CA. Not all of these certificates are always required, as they can be exchanged in the initial EAP-TLS session.

We note here that the configuration of the end user device is not embodied in a downloadable executable, or in a vendor-specific configuration file. Instead, the configuration is given in the form of standardized certificates which have been marked up to express their intended usage.

When the end user receives the client certificate (and possibly certificate chain), it can be installed on the device immediately. When the certificate is installed, it causes a number of additional steps to be taken by the device which is using that certificate. These steps are new to EAP peers, and are not performed today.

First, the device determines that this certificate contains `id-kp-eapClient`, which indicates that this certificate is intended to be used for EAP. If there is a certificate chain, then those certificates can be saved. If the client certificate contains `id-pe-wlanSSID`, then the device uses that information to configure itself so that it will connect to the named SSID, and to perform EAP-TLS authentication using this client certificate.

This process is similar to that outlined in [RFC4334]. The changes we make over that specification are new extensions to the certificates, and additional steps which mandate new behavior.

Finally, if there is a root CA in the certificate chain, that the root CA is installed. The device also annotates this root CA (pre-existing or otherwise) as being trusted to issue certificates for use with EAP. As we will see, the device can also require the EAP server certificate to contain `id-kp-eapServer`, along with an `id-on-naiRealm` value which matches the `id-on-naiRealm` which is in the client certificate.

At this point, the end user device is fully configured for using EAP-

TLS with a particular SSID.

5.1.4. Obtaining Network Access

When the end user device wishes to obtain network access, it can for the most part follow the methods used prior to the publication of this specification. There are, of course, a few changes which simplify the process and make it more secure.

For privacy, the device uses an anonymous identifier in the EAP Response Identity field. This identifier is the NAI realm which is taken from the id-on-naiRealm field of the client certificate. Taking the NAI realm from the client certificate means that there is no need for the user to configure the publicly visible EAP Response Identity. This usage also provides for the anonymity required in [EAPTLS].

In order to provide for privacy of the client certificate, TLS 1.3 is used. Older versions of TLS are NOT RECOMMENDED.

When the EAP-TLS connection is established, the device verifies that the server certificate which is presented also contains a id-on-naiRealm field, which matches the value in the client certificate. This validation is similar to the validation of DNS names performed by web browsers when accessing HTTPS sites. However, as DNS is not available during EAP authentication, the id-on-naiRealm field is used instead to validate the server certificate.

For clarity, we repeat the instructions in [RFC7585] Section 2.2, for matching the NAI realm:

The comparison of an NAIRealm to the NAI realm as derived from user input with this algorithm is a byte-by-byte comparison, except for the optional leftmost dot-separated part of the value whose content is a single "*" character; such labels match all strings in the same dot-separated part of the NAI realm. If at least one of the sAN:otherName:NAIRealm values match the NAI realm, the server is considered authorized; if none match, the server is considered unauthorized.

Since multiple names and multiple name forms may occur in the subjectAltName extension, an arbitrary number of NAIRealms can be specified in a certificate.

The device also verifies that the server certificate also contains the id-kp-eapServer field. It verifies that the certificate is signed by a root CA which is annotated as being permitted for use with an EAP server. If these verification steps fail, then the

client stops the authentication process, as it has determined that the network is not trusted.

If all of these verification steps pass, then the end user device can trust the EAP server, and be authenticated to the network.

We note here that from the perspective of the end user, the only actions which have been performed have been (1) to install a certificate, and (2) to enter a password for that certificate. This process is substantially simpler than most WiFi configuration processes used today. This process is also likely to be easy to follow for most users.

5.2. Other TLS-based EAP methods

While the above example discusses EAP-TLS, it is easily extensible to any other TLS-based EAP methods. Instead of distributing a client certificate to end users, the administrator can distribute a server certificate and/or a root CA which is intended for use with EAP. For simplicity, the server certificate should also contain id-pe-wlanSSID, which informs the client as to which SSID(s) should be used for authentication.

We reiterate that the public EAP Response Identity used should always be in the form "@realm", as per [EAPTLS] Section 2.1.7. The user's full identity should only be sent inside of the TLS tunnel. We also recommend that the inner authentication methods use the full identity of "user@realm", and not just the "user" portion.

The end device then follows the same process to configure the SSID for authentication, to mark up the SSID as being used with a particular NAI realm, and to annotate the root CA as being permitted for use with EAP. Again, having an SSID here simply makes this example clearer, this specification does not mandate its use, and this specification is applicable to any type of network access which uses EAP.

When the device connects, it does not need to verify that the server certificate being presented is the same as was used for configuration. Instead, the device simply has to "pin" the combination of SSID, NAI realm, and root CA. This pinning allows for flexibility in accepting other server certificates, while preventing down-grade attacks which attempt to supply different root CAs for that NAI realm. This pinning means that the device associates the SSID and NAI realm with a particular root CA, and then does not permit that NAI realm to authenticate to an EAP server which does not use the same permitted root CA.

The only requirement on the new server certificate is that it has to match the same criteria as outlined in the previous section. That is, the certificate must contain `id-kp-eapServer`, it must have a matching `id-on-naiRealm`, and it must be signed by a root CA which the supplicant has permitted to be used with EAP.

The device can then safely send authentication credentials inside of the TLS tunnel. This process is substantially similar to that used to log into an HTTPS enabled web site. The only difference here is that the device must associate the user's credentials with EAP and an NAI realm, instead of with the web and a DNS host name.

5.3. EAP methods which do not use TLS

Unfortunately, the methods outlined here apply only to TLS-based EAP methods. This limitation is because we are leveraging the TLS certificate format in order to both define additional permitted uses for those certificates, and to inform devices how non-TLS systems should be configured.

This extra information is simply not possible to add to other EAP methods such as EAP-PWD [RFC5931]. Those methods typically authenticate users, but do not provide for carrying additional information. Those methods also generally do not provide for the mutual exchange of identities, and for mutual authentication.

Authentication protocols such as EAP-PWD are simple enough that it is both impossible, and generally unnecessary, for them to use the methods outlined in this specification. That is, the cryptographic guarantees in EAP-PWD ensure that it is always safe to perform EAP-PWD with unknown EAP servers, as there is no possibility for leakage of user credentials. As such, there is less need to verify the identity of the EAP-PWD server.

5.4. Other Methods of Provisioning

The EAP-TLS example described how provisioning was done via an administrator sending certificates to an end user. This process is not always necessary.

For EAP-TLS, [EAPTLS] Section 2.1.5 provides for the protocol to be used without peer authentication. This capability can be leveraged to perform provisioning. All that is needed on the device is for the EAP peer to have a pre-configured root CA, and to know the NAI realm which it belongs to, and which is being used for provisioning.

For the purposes of this section, we assume that the root CA known to the device is willing to issue and sign server certificates which

contain the id-kp-eapServer and id-on-naiRealm fields. As we will see below, this assumption may be difficult to achieve in practice.

When the device connects to a network, it can perform the verification steps outlined above. That is, the server certificate presented has a matching id-on-naiRealm field; the server certificate contains id-kp-eapServer; and that the server certificate is signed by a known root CA. The root CA does not necessarily have to be annotated as being permitted for EAP.

If all of those requirements are satisfied, then the device can obtain limited network access. The device can then leverage normal networking protocols to download provisioning information, which is then used to configure the device. As noted above, this provisioning information needs to be little more than a client certificate.

For example, the device could use Enrolment over Secure Transport (EST) [RFC7030]. It could also use vendor-specific methods.

This process works because every device capable of doing TLS is shipped with a set of known root CAs, which are intended for use with the web. In addition, every end user wishing to connect to a known network is aware of the identity of that network (e.g. "example.com"), and their identity in that network (e.g. "user@example.com").

If the device does not have a root CA configured, as we will see below, it can use the limited authorization network with other protocols such as DNS. The device could use DNS to query a pre-defined SRV record (as with [RFC7585] Section 3). The results of that record could be a "self signed" root CA. Certificates can therefore be obtained over DNS, such as via the methods outlined in [RFC4398].

The only requirement here is that the DNS record be obtained securely (DNSSec or DNS over TLS), otherwise an attacker could forge the response, or replace the root CA in transit.

Other methods are also possible, though not discussed here.

5.5. Trust on First Use Can be Secure

Similar provisioning methods can be used for other TLS-based EAP methods. In those methods, when a device connects to the network, it could prompt the user for a username (with an NAI realm) and password. Then, it could use that information to derive the NAI Realm, and perform the verification steps described previously. The device simply needs to know that it is trying to authenticate to a

specific NAI Realm before verifying the server certificate, and needs to verify that server certificate prior to sending any user identity or authentication credentials to the EAP server.

That is, a device which knows that it is trying to authenticate to a realm "example.com", can then verify that the server certificate contains id-on-naiRealm which matches "example.com". This process is similar to a web browser that wishes to connect to a web site for "example.com". The client already knows that "example.com" is the desired destination, which then means that the client must verify that the site which it connects to has a certificate matching "example.com".

If the device is not configured with any realm, then it has no way of determining whether or not it should trust any EAP server. As such, the use of the NAI realm is a critical component of this specification.

This process is close to "Trust on First Use" (ToFU) provisioning, with minimal knowledge required, and with a high degree of security. From the point of view of the end user, the only actions which have been taken are to select an SSID, and then to enter a name and password. The process outlined here ensures that the user is authenticated to a known and trusted network, and that the EAP peer sends the identity and authentication credentials only to known and trusted networks.

Some EAP methods such as TEAP [RFC7170] support provisioning of end user devices. Since this provisioning information is automatic, it can include additional information not discussed here. The process, however, remains substantially similar. The client can download one or more certificates, and then perform the validation and configuration steps outlined above.

5.6. Additional Considerations

Note that there is no requirement that the device use only the SSID given in the id-pe-wlanSSID field of a certificate. If the device sees an authorized server certificate on a different SSID, then it should proceed with authentication as discussed previously.

However, the EAP server may not permit the client to be authenticated via other SSIDs. It is therefore RECOMMENDED that channel bindings [RFC6677] are used in all EAP methods, in order to inform the server about the clients local environment. Channel bindings also solve problems with supplicants that do MAC address randomization: The real MAC address is sent inside of the TLS tunnel, as part of the channel binding exchange.

Similarly, there is no requirement for the device to "pin" a particular server certificate. If the presented server certificate meets the criteria of known root CA; containing id-kp-eapServer; and matching id-on-naiRealm, then the connection can be trusted. In fact, there is no need to cache the server certificate at all.

6. Extending the Solution

The process described above greatly simplifies the usability of EAP, and its security. We can, however, do better.

The process described above requires changes to both supplicants, and to the systems which issue certificates. These changes are useful, but are not always trivial. Further, the processes still have a bootstrapping problem, which was waved away in Section 2.2.3 during the discussion of the worked example. The bootstrapping problem was somewhat addressed by the use of ToFU provisioning in Section 2.6, but there are still open issues with respect to security and provisioning.

In this section, we describe a few ways in which the remaining issues may be addressed, in order to come up with a complete solution to the problem. We first describe protocols such as EST [RFC7030], and then later how DNS may also be used.

6.1. Bootstrapping via EST

EST [RFC7030] can be used to distribute previously created certificates for CAs and servers. It can also be used by clients to request new client certificates. As there is no distinction in EST between public CAs and private CAs, either can be used. This feature enables EST-capable systems to use the new OIDs defined here.

The use of EST therefore solves the certificate distribution problem which was described earlier in the worked example for EAP-TLS.

However, the requirement here is that the client implements EST, and not all currently do. Another requirement is that EST uses the ".well-known" relative URL from [RFC5785], and both specifications assume implicitly that the base domain is rooted both in the web, and in the top-level subdomain. For example the base URL for "example.com" is given as "www.example.com". While the ".well-known" prefix is capable of being used with other portions of the domain name tree, there is no standard way for a client and server to agree on its location. The location is simply implicit in the protocol.

This limitation is an issue mainly because we wish to use automatic enrollment schemes in "captive portal" or "walled gardens", which have limited network access. It may be possible for a system with limited network access to reach a URI such as "https://www.example.com/.well-known/". However, there is no guarantee that the administrator of the main web server system for a domain is the same as the administrator of the captive portal system.

As a result, we need a way for both the client to discover the URI to use, and for that URI to point to a web sever which is controlled by the administrator of the captive portal system. The solution here is to use DNS.

We define a DNS SRV record which points to the EST server for this NAI realm. In order to provide for the separation of responsibilities, we make this record specific to EAP.

The format of the SRV record is as follows:

```
_est._eap.<naiRealm>
```

An EAP client system can query for this record, and then connect to the ".well-known" service at the provided host, in order to perform EST. This record may point to the main EST server for a domain, or it may point to a separate EST server which is specifically used for EAP. We believe that it is important to make provisions for separation of services such as these, even if this separation is not always used.

We note that the DNS resource record type here is SRV and not URI, as we only need to define the hostname and port for the EST server. The rest of the URI is defined by EST.

We discuss the use of DNS in more detail in the next section.

6.1.1. Closing the loop

An EAP peer which has a username, password, and NAI realm can leverage EST to securely provision client certificates for use with TLS-based EAP methods.

The device first discovers the location of the EST server via the DNS lookup above. It can then download any necessary CAs, using the methods outlined in [RFC7030] Section 4.1 The device then uses the username and password with HTTP basic authentication in order to authenticate itself to the EST server. Finally, the device can request that the EST server create and sign a client certificate, using the methods outlined in [RFC7030] Section 4.2.

One benefit of this method is that the key associated with the client certificate can be generated automatically by the client device, and to some extent hidden from the user. This secrecy ensures that the client certificate is associated with a particular device, and that the user is prevented from copying the certificate to multiple devices.

6.2. Bootstrapping via DNS

We have seen above that the EAP configuration problem can be largely reduced to getting a properly formed certificate onto the device. We show here how to use DNS to bootstrap the certificate installation. This bootstrapping process requires no changes to supplicants or to systems which certificates. Instead, it requires only that:

- * certificates be placed at a well-known URI,
- * this URI is found DNS CERT records [RFC4398],
- * an independent tool downloads the certificates and uses them to configure EAP as described above,
- * the end user or device knows the NAIRealm to which it is supposed to connect.

This process leverages both DNS, and the existing "web" root CA infrastructure in order to securely configure EAP, with minimal manual intervention.

6.2.1. CERT records

The process begins with the administrator obtaining one or more server certificates as described above, and then placing them at a well-known URI. The administrator then adds records to DNS which point to the certificates. The client can then download these server certificates, and configure its EAP system to use these certificates when authenticating to the relevant NAI realm.

For the purpose of this section, we assume that the server certificates are signed by a CA which is already known to the client system. In the next section, we extend this process to downloading new CA certificates.

The information stored in DNS is a CERT record as described in [RFC4398] Section 2. If the DNS record is served over a secure transport such as DNSSEC, DNS over HTTPS, or DNS over TLS, then the record can directly contain the certificate. If the DNS record is served over an insecure transport, then the "type" field MUST be one which contains a URI. These requirements typically mean that value of the "type" field will be (4), for IPKIX, which points to the URL of an X.509 data object.

In order for this process to be secure, the URL MUST be within same domain (NAI realm) as the CERT resource record. The URL MUST be secured with TLS transport. The certificate presented at that URL

MUST be issued by a root CA which is generally already known to the device. The certificate presented at the URL MUST pass all normal HTTPS validation, including that for id-ce-subjectAltName.

That is, when the client accesses a URL pointed to by a CERT record, certificate validation for that access MUST be performed as per [RFC5280]. If any of these validation steps fail, then the client MUST NOT download or use any further data presented by that server.

Further, the contents of the data at that URL MUST be a X.509 certificate.

The downloaded certificate MUST be one with is suitable for TLS-based EAP methods, as described in [EAPTLS]. The client system MUST verify that the server certificate matches the NAI realm which is being used, either via the steps defined here, or via the host name matching defined in [EAPTLS]. If the certificate does not match the NAI realm, then it is discarded and not used for EAP.

An issue is that [EAPTLS] provides for host name matching, but not for NAI realm matching. The two are similar, but not identical. If we recall [RFC7542] Section 2.5, the NAI realm is defined as:

* Realms MUST be of the form that can be registered as a Fully Qualified Domain Name (FQDN) within the DNS.

Certificates of the form supported by [EAPTLS] therefore may still match the given NAI realm.

The downloaded certificate SHOULD also contain id-pe-wlanSSID, in order to inform the device as to which SSID is suggested for network access.

Note that there is no requirement for this server certificate to contain the id-kp-eapServer OID defined here. It is RECOMMENDED to include that OID, but it is not required.

These requirements ensure that devices can leverage the existing web framework to securely download certificates which are to be used for EAP.

The use of insecure transport for DNS is acceptable, as it is only being used to transport a URL, which is itself protected by TLS. The URL validation requirements above ensure that an attacker can only point the device to pre-existing URIs within the given domain, which contain information not under the control of the attacker.

6.2.2. CERT record labels for Server Certificates

The next problem is to define a well-known name for this record. We leverage the [RFC8552] "Underscored" naming of attribute leaves in order to provide for well-known names. We define a series of names, which are all rooted from the NAI realm given by the user.

We note here that the insecurity of plain UDP DNS may, in fact, be of use here. For example, the administrator of a captive portal can modify the captive portal DNS server in order to serve records for the "top level" domain, which not normally be permitted. Since this use of DNS names is only visible from within the captive portal, there is no security impact outside of this limited network.

The format of the CERT record is as follows:

```
_server._cert._eap.<naiRealm>
```

It can be beneficial to use a DNS CERT record instead of Enrolment over Secure Transport (EST) [RFC7030], as our goal here is to simply obtain a pre-existing certificate, and not to generate new certificates. In some cases, the URL provided by DNS can just be the URL of a certificate hosted by an EST server.

In some cases, the downloaded certificate may be from a CA which is not known to the device. For example, when the CA is a "private CA" which is not in the root CA list for web PKI. The next section addresses this issue.

6.2.3. CERT record labels for CA Certificates

This CA certificate may be obtained via EST, as described in [RFC7030] Sections 2.1 and 4.1.2. The device can also look up the CA certificate via a similar process to obtaining the server certificate, by querying for a CERT record at the following name:

```
_ca._cert._eap.<naiRealm>
```

Again, the URL presented here MUST match all of the requirements given earlier for the certificate obtained from the "_server._cert._eap.<naiRealm>" record. The only restriction on the contents and/or format of the downloaded CA certificate is that it MUST permit the previously downloaded server certificate to be verified. If the server certificate cannot be verified using this CA certificate, then both certificates MUST be discarded.

Since this new CA has been downloaded from a trusted source, the CA can also be given limited trust. That is, the downloaded CA SHOULD

be trusted to issue certificates for use with EAP, but only for the NA realm in question. The downloaded CA MUST NOT be trusted for any other use-cases or purposes. This limitation ensures that private CAs cannot be used to spoof public web sites from unrelated organizations.

This requirement in effect mandates implementations to create multiple certificate stores. This limitation is the minimal change required to supplicant implementations in order to support the core of this specification. Every other change suggested here can either be pushed to an auxiliary tool, or can be delayed until a later step.

That is, the user-visible workflow here can be implemented with minimal changes to the supplicant software which implements EAP.

7. Related issues

We discuss related issues in this section. The items discussed here are individually useful to discuss, but do not follow a clear developmental flow. As such, they are placed into a separate section.

7.1. Provisioning Issues

There are a number of issues related to provisioning. We show that there is no need to use a single network for all of the above discovery and configuration. We show that configuration updates are simple, and are no more difficult than repeating the initial provisioning. Finally, we describe why the methods defined herein are significantly more secure than ToFU.

7.1.1. Bootstrapping via a Separate Network

There is no requirement that a particular network provide all of the bootstrapping outlined above via a "guest network". It is also possible to leverage the public Internet in order to bootstrap authentication to a private network which requires EAP authentication.

For example, a mobile phone may be trying to connect to a WiFi SSID, while it also has additional network access via 3G or LTE. There is no requirement here for the WiFi network to provide a guest network with full provisioning capabilities. Instead, the phone can simply try to do unauthenticated EAP-TLS. During the EAP-TLS negotiation, the device will obtain a copy of the server certificate. This certificate should contain id-on-naiRealm.

The device can then use the LTE connection, and the process outlined

above (DNS, EST, etc.) in order to verify that the server certificate is the one which meets all of the criteria necessary for full authentication. Once the server certificate is validated and the device has updated its configuration, it can drop the EAP-TLS connection, and re-authenticate using any TLS-based EAP method.

With this process, the experience for the end user would be little more than:

- * select an SSID,
- * be informed that this is a network associated with a particular NAI realm (i.e. domain),
- * be informed that the network is secure and is trusted,
- * be requested to enter an identity and password within that domain,
- * enter that identity and password, and obtain network access.

This process is little different from using a web browser to navigate to a web site, and ensuring that the green "lock" icon is set for that site.

7.1.2. Configuration Change is just Refresh

Any automatic provisioning scheme has the problem of performing change control. In our case, updating the configuration with a new set of data is largely just repeating the bootstrapping process. The questions then become how often to check for updates, how long to cache configuration, etc.

It is RECOMMENDED that HTTPS servers which provide the certificates described in the previous section set the Cache-Control [RFC7234] directive in the response. It is RECOMMENDED that the "max-age" directive ([RFC7234] Section 5.2.2.8) be used. The value returned SHOULD NOT be less than one day (86400 seconds), and MUST NOT go past the expiry date of the certificate which is being returned.

The supplicant which is retrieving the certificate SHOULD annotate the certificate with the value of the "max-age" directive. The supplicant SHOULD perform the bootstrapping checks again prior to the "max-age" time limit being reached.

Where "max-age" is not returned, the supplicant SHOULD refresh the bootstrapping checks again no more than once per day. It SHOULD track when the certificate was downloaded, and then perform these checks no later than when the certificate is halfway to expiry, taken

from when the supplicant first downloaded the certificate.

When either the root CA or server CA has expired, the supplicant **MUST** NOT use them to obtain network access. It **SHOULD** refresh the certificates at that time. If the certificates are not refreshed, then the relevant configuration **SHOULD** be deleted.

If the refreshed certificate is identical to the previously downloaded certificate, then the supplicant makes no configuration changes other than to update its refresh timers. The supplicant **SHOULD** still perform other certificate validation checks, such as checking for certificate revocation.

If the refreshed certificate has changed, then the supplicant performs all of the validation checks described above. If the tests pass, the new certificate can be used in place of the previous one. Note that there is no need to "tear down" the current network connection if the current certificate is still valid. The new configuration should be used only when the device next requests network access.

As the old credentials are usually still valid, device **SHOULD** keep the old credentials around until such time as it has verified that the new credentials work. If the new credentials do not obtain satisfactory network access, then they should be discarded, and the device should try again not sooner than one day later.

TBD: There should also be fine-grained methods to control when a new configuration is downloaded, and separately when it is applied. For client certificates, we can use the "notBefore" field, which indicates that the certificate is not valid before a particular time.

7.1.3. Secure versus Insecure Provisioning

We now revisit the discussion of ToFU first mentioned above. We note that the process defined here isn't even "trust on first use". Instead, it is leveraging the web PKI in order to get secure, authenticated downloads of non-web certificates. ToFU provision such as used in TEAP is essentially a standardized way to download security configuration from an insecure source.

Our proposal here begins with the naiRealm, and then uses trusted roots and secure protocols to download security configuration from a known and trusted source. While this process is more complex than TEAP, in that it requires DNS and HTTPS, it is also more secure.

7.2. Issues related to Security

We explain why id-on-naiRealm was chosen. We describe some issues related to resumption, and the use of certificates in a multi-server environment. We explain how this solution can be extended to configure individual EAP types. We explain how this solution is applicable when either private or public CAs are used. We conclude by explaining how the user experience offered by this solution creates a simple and clear user experience.

7.2.1. Why id-on-naiRealm

Server certificates used with EAP have historically contained DNS names. This practice is largely because the certificates are "TLS web server" certificates. However, [RFC7585] Section 2.2 explains why DNS names are not appropriate:

Current subjectAltName fields do not semantically allow an NAI realm to be expressed; the field subjectAltName:dNSName is syntactically a good match but would inappropriately conflate DNS names and NAI realm names. Thus, this specification defines a new subjectAltName field to hold either a single NAI realm name or a wildcard name matching a set of NAI realms.

We extend the above requirement to say that the wildcard name MUST be limited to a subset of one realm. That is, a wildcard of "*.example.com" is permitted, but a wildcard of "*", or "*.com", is forbidden.

Although this recommendation was done in the context of RADIUS, this field is exactly what is needed for EAP. The definition is the same (NAI), and the use-cases are the same.

7.2.2. Resumption

[RFC8446] Section 4.6.1 discusses resumption:

Clients MUST only resume if the new SNI value is valid for the server certificate presented in the original session and SHOULD only resume if the SNI value matches the one used in the original session. The latter is a performance optimization: normally, there is no reason to expect that different servers covered by a single certificate would be able to accept each other's tickets; hence, attempting resumption in that case would waste a single-use ticket. If such an indication is provided (externally or by any other means), clients MAY resume with a different SNI value.

-0.3i

Similar requirements apply for EAP, except that clients check id-on-naiRealm instead of using SNI.

Where multiple servers are in an high availability or load-balance group, they SHOULD use the same certificate. Where the same certificate is used, then either the resumption master secret MUST be shared among all systems, or the tickets MUST be accessible to all systems. Preferably by putting them into an external data store.

7.2.3. Choosing EAP Types

We note that this specification does not define which EAP type is used by the supplicant, except implicitly. That is, if the supplicant is given a client certificate, then it is presumed that EAP-TLS is being used. Otherwise, the supplicant should choose some other TLS-based EAP type.

It would be possible to define new OIDs which define a list of EAP types that the EAP server will accept. These OIDs can then be placed in a server certificate, where they can inform the supplicant as to which EAP types should be used.

7.2.4. User Experience

It is RECOMMENDED that the system notify any end user of the configuration changes being performed. It is RECOMMENDED that these notifications give sufficient information to the end user, so that an informed decision can be made. It is RECOMMENDED that these notifications allow the user to stop or cancel the process at any time.

The goal of the user experience described is that it should be little different from using a web browser to navigate to a web site, and ensuring that the green "lock" icon is set for that site.

It is therefore RECOMMENDED that supplicant vendors update their user interfaces to clearly distinguish between "trusted" and "untrusted" network access. A "trusted" network is one which satisfies all of the criteria outlined herein. An "untrusted" network is one which satisfies only some, or none of the criteria outlined here.

It is likely a good idea to update the graphical user interface (GUI) for the supplicant with a green lock / red unlocked icon, similar to that used in web browsers. Further, the GUI should also include the naiRealm which has been verified, as web browsers show the domain name which has been verified. This information is enough to give the user enough information to meaningfully consent to obtaining network

access, and to enter credentials.

That is, if the user sees that the operating-system GUI window says "this site is trusted", and also "you are accessing the example.com domain", then the user can safely enter credentials for that domain.

This workflow is familiar to end users, and has been proven to be at least moderately successful in the web.

7.3. Issues related to Certificates

There are a number of other issues related to certificates, in addition to those which have been raised above.

TBD: more explanation of the trailing sections.

7.3.1. Public CA versus Private CA

Nothing in this specification requires the use of either a public or private CA. Both are possible, and both have issues.

The main issue with using a private CA is that it is not already on the device, and has to be provisioned. While there are many possible methods of provisioning this information, we define here only a few straightforward methods. We hope that the method proposed here (DNS + HTTPS) is clear, and simple for administrators to implement.

There is still the requirement that the client device have new software to obtain this information and call the supplicant API. However, this process is no different than installing custom MDM software.

Private CAs have the benefit of being able to sign certificates with any EKU they desire. These certificates can then be marked with an EKU as being intended for a particular use, and supplicant software can verify these EKU fields.

The issues with public CAs are described above. Public CAs are likely to refuse to sign certificates which contain the EKUs proposed here, and appear to be uninterested in offering a different product which would sign such certificates. Further, using these certificates for EAP appears to be against their intended purpose, and is therefore misuse.

However, the benefit of using public CAs is that they are already configured on most devices, and it is relatively simple to obtain certificates from them.

In the end, local administrators can choose whatever CA is best for them. Our goal here is to simplify the process of using a CA and server certificate for EAP. It is best to give administrators and implementors a few simple options which meet their needs, rather than mandating one particular solution which is likely to not meet the needs of a large set of users.

7.3.2. Limitations of public CAs

Recent changes to the CAB guidelines limit certificate validity periods to 397 days. While this change may be good for the larger WWW framework, it is not clear how it benefits other protocols such as EAP. Additional changes include limiting the kind of domain validation methods permitted, and forbidding file-based validation for wildcard certificates.

Part of EAP "best practices" is to ensure that EAP (or AAA) servers have minimal exposure to the public Internet. In order to use certificates from a public CA therefore, administrators must choose between either exposing their EAP server via WWW (in order to perform validation), or to expose a different WWW server, and then also simultaneously install the same certificate on the EAP system. Neither option is a good one.

In addition, limiting the certificate validity period means that clients see the server certificate change much more often than has been previously the case. This higher volume of changes was historically perhaps not an issue, as we have seen above that some systems perform limited checks on server certificates.

The benefit of the process outlined here is that the server certificate either becomes trivially verifiable (even if it changes), or installing a new server certificate becomes a trivial and secure process.

On the other hand, if there is no automated process to update the EAP client configuration, then users will simply be trained to mindlessly click "accept" when they are presented with a new certificate. As this process will happen much more often than has historically been the case, this maladaptive behavior by users will be even more strongly enforced.

Another issue with public CAs is that intermediate CA certificates are significantly more expensive than a server certificate. The reason here appears to be economic: if intermediate CA certs were cheap, then an organization would simply purchase one, and then use that CA cert to issue many server certificates.

This limitations means that EAP-TLS is significantly more difficult to deploy in practice and PEAP or TTLS. The administrator has to choose between either purchasing an extremely expensive intermediate CA certificate, or using a private CA.

The use of intermediate CAs has other issues, as we will see in the next section.

7.3.3. CA Chains

Some administrators wish to use multiple CAs for security. For example, a large organization could have one CA which is controlled by a security group. That CA could issue intermediate CA certificates to other groups with the organization. These CAs could be issued on multiple grounds, such as geographic location, or function units.

In practice, this process could result in there being one CA which is used for EAP / AAA, another CA which is used for internal web sites, and another CA for organizational Virtual Private Network (VPN) usage.

We have worked through all of the examples and discussion above by largely assuming that there was one "root" CA. Now that we see this assumption is insufficient, we must then discuss, and solve, the issue of intermediate CAs.

If an application were to simply trust the "root" CA, then by inference it would also trust all intermediate CAs. This trust means that an EAP administrator who can issue client certificates could potentially configure that client certificate for use in another protocol, such as with a VPN. Such misuse could lead to unauthorized users obtaining access to resources.

The solution here is two-fold. One, applications which accept client certificates SHOULD be configured to trust a particular issuing CA, which may not be the "root" CA. That is, simply having a certificate store of root CAs is insufficient. Instead, the application needs to track a particular intermediate CA.

Another solution is to simply move to purpose-specific EKU fields. An EAP client which follows this specification can require that the EAP server contain an id-kp-eapServer field. The EAP client can then rely on policy within the issuing framework to ensure that all relevant certificates also have an id-kp-eapServer field. Similarly, and EAP server which follows this specification can ensure that EAP clients are presenting certificates which contain id-kp-eapClient.

That is, the use of id-kp-serverAuth for all possible applications means that it is impossible to limit the use of certificates to one particular application. An administrator or end user is free to (mis-)use any certificate for almost any purpose.

We would suggest that having purpose-specific key usage fields is preferable. Such fields would make it simpler for both clients and servers to have more fine-grained control over certificate usage.

7.3.4. Delegated Authentication

In some cases an organization may delegate EAP / AAA functionality to another organization. This can happen for example, when an organization does not wish to run authentication servers itself, but instead delegates that functionality, say to an identity provider (IdP). The delegated functionality may be operated by an organization which handles "authentication as a service" for multiple customers.

A current solution is for the IdP system to present a server certificate which contains a list all of the domain names which it services. The problem is that this list can change often, which means that the old certificate must be revoked, and a new one issued. If these changes happen regularly, then this "churning" of certificates can cause problems for clients which cache the server certificate. There is also the management overhead of updating the certificate. Over all, this process is not scalable.

The processes outlined here allow for simple discovery and configuration of TLS-based EAP methods, but they do not entirely solve this problem.

The problem can be solved, however, by noting that the public EAP Response Identity used should be in the form "@realm", as per [EAPTLS] Section 2.1.7. An EAP server will receive this identity in the first EAP packet, at which point the server can select and present a certificate which is appropriate for that realm.

The result is that an IdP needs to be configured only with one server certificate for each NAI realm that it manages. When an NAI realm is added, deleted, or updated, those changes affect only the configuration for the modified realm. Any other organization or NAI realm is not affected.

This solution is simple and scalable.

7.3.5. Identification of Networks

While the examples above used an SSID to identify a network, there are other ways of network identification.

One is the Roaming Consortium Organisation Identifiers (RCOI), which are organizational identifiers which are assigned by the IEEE (REF TBD). They can be 24 or 36 bits. These organizations are global, and can identify a vendor, operator, consortium, or other organization.

This section defines the RCOIdentifier name as a form of otherName from the GeneralName structure in subjectAltName defined in [RFC5280].

```
id-on-RCOIdentifier OBJECT IDENTIFIER ::= { id-on TBD }
```

```
ub-RCOIdentifier-length INTEGER ::= 255
```

```
RCOIdentifier ::= OCTET String (SIZE (1..ub-RCOIdentifier-length))
```

This field can be used in either a client certificate or a server certificate. With either usage, it indicates to the client which RCOI should be used for accessing network services.

7.4. Anti-solutions

In this section, we explain why a number of existing technologies do not solve the problems which are addressed by this specification.

7.4.1. MDM Products Are not the Solution

MDM products are not the solution. Solutions like Eduroam CAT [CAT] are simple and easy to use, but they are only one of many possible products. In the extreme case, each end user has to download one MDM product for each network being accessed, and then repeat that process across each of many devices being used to access those networks.

These solutions are not just expensive, and non-standard, they are not scalable. It is difficult to scale the solutions to millions of disparate devices, as software has to be written and verified for each vendor, and often for each firmware version supplied by a vendor.

In addition, MDM products do not scale for an individual device. Each MDM product usually assumes that it is in complete control of the device, which makes it difficult or impossible to install multiple products. For example, a contractor who works for multiple companies may need multiple conflicting MDM products. Or, an

employee may be required to install an MDM product on a personal device, which makes it difficult to say who actually owns that device.

These MDM products typically also are capable of remotely wiping the device, such as when a contractor or employee leaves an organization. If the device was bought for personal use, there are ethical and may also be legal implications. Other issues are the loss of critical data such as documents or personal photographs.

Or perhaps even worse, when an MDM product is in complete control of a device, then there is plausible deniability for a user, for any action taken on that device. It is likely defensible to claim that the user is not responsible, because "the remote admin had full control", or perhaps "the remote admin is running software which controls my device".

There are serious security issues with a user not being in control of their own device.

In contrast, a standard discovery and configuration method, run by devices at the edge, which leverages DNS and HTTP is proven to work at Internet scales. They are implemented once by each vendor, and then maintained afterwards. As the configuration for each organization (NAI realm) is separate, there are minimal issues with installing multiple configurations on the same machine.

However, in the interest of enabling multiple solutions, we also define an [RFC8552] URI record. This record points to a location where a client device can download an MDM solution which is specific to a particular organization.

The format of the URI record is as follows:

```
_install._mdm.<naiRealm>
```

This record SHOULD NOT be visible on the public Internet, i.e. the public DNS servers for that NAI Realm. Doing so would permit malicious actors to download and examine the MDM software. Instead, this record SHOULD be available only inside of the organizations private network. Either to devices which have used 802.1X in order to authenticate themselves to the organization, or to devices which are using private IP address ranges.

The server which hosts the URI SHOULD use device fingerprinting in order to provide a system-dependent MDM solution.

As with the requirements on certificates above, the URI MUST be

within same domain (NAI realm) as the CERT resource record. The URI MUST be secured with TLS transport. The certificate presented at that URI MUST be issued by a root CA which is generally already known to the device. The certificate presented at the URI MUST pass all normal HTTPS validation, including that for id-ce-subjectAltName.

If any of these validation steps fail, then the client MUST NOT download or use any further data presented by that server.

If the validation steps succeed, then the client device can download and run the MDM software which has been provided at that URI.

Where possible, the client MUST inform any human user that these steps are being taken, and MUST give the user the ability to prevent this download from happening. There are many situations where the client device is owned by the end user, and not the organization which is being accessed. As such, it is inappropriate to mandate that software be automatically installed.

However, there is also no requirement that an organization grant access to devices which do not follow the organizations policy. The organization is free to deny the device network access until such time as the MDM software has been installed.

7.4.2. EST and similar protocols do not solve all of the problem

Certificate provisioning solutions like EST [RFC7030] or Simple Certificate Enrolment Protocol (SCEP) [RFC8894] are useful, but they do not solve the underlying problem we solve here.

EST and SCEP are useful for provisioning CA certificates to end devices, and for end devices to request and provision client certificates. However, these processes generally require additional configuration on the client device, and also an unrestricted network connection.

Part of the problem we are trying to solve here is supplicant configuration, and EST / SCEP do not help.

Further, EST can require complex bootstrapping. Section 2 of [RFC7030] says:

Both the EST clients and server are configured with information that provides the basis for mutual authentication and for authorization. The specific initialization data depends on the methods available in the client and server, but it can include shared secrets, network service names and locations (e.g., a Uniform Resource Identifier (URI) ...), trust anchor information

(e.g., a CA certificate or a hash of a TA's certificate), and enrollment keys and certificates.

In contrast, the method proposed here requires that the client device have a known root CA from the web PKI, and the ability to do DNS and HTTPS. This capability is available on essentially all systems which can access the public internet.

The reason for this simplification is that the problem we are trying to solve for EAP is substantially smaller than the problem that EST is trying to solve.

We conclude by noting that our solution is entirely compatible with EST, in that the DNS query for "_ca._cert._eap.example.com" could return a CERT record which points to the URL of the EST server, for example "https://example.com/.well-known/est/cacerts", as described in [RFC7030] Section 4.1.2.

7.4.3. Captive Portals and Hotspots

Captive portals and hotspots have been traditionally used as a method of controlling network access, as with EAP. The use-case for captive portals is that the client devices can do DNS and HTTP, but that they do not have credentials already provisioned. The captive portal is usually a way to introduce humans into the process, by displaying information about the network, and asking the user for credentials such as credit cards, etc.

There are, of course, a number of issues with captive portals. It may take the client device some time to determine that it is in a captive portal. The information displayed on a captive portal page may be confusing to the end user, or may even be in a language which the user does not understand.

Automating the onboarding process means that almost all of these issues are resolved.

7.4.4. Fully Anonymous Network Access and Provisioning

In some situations, the device may have minimal authentication credentials. There is still, however, a need to provision the device.

Existing solutions such as [RFC7170] Section 3.8 rely on exchanging data inside of a tunneled EAP method. This process is useful, but does not permit either device to use the full suite of protocols available on the wider Internet. We therefore define a different provisioning method here.

When devices need to provision themselves, they SHOULD attempt to authenticate using EAP-TLS, with no client certificate, and an NAI realm of "@eap.arpa". This realm indicates to the server that the device wishes to perform provisioning.

In some situations, the client device can authenticate the server certificate presented by the EAP server. More generally, however, it cannot. The client device therefore SHOULD NOT attempt to verify the server certificate when using this provisioning method. The device MUST treat the network as untrusted, and the server MUST ensure that the device is placed into a "captive portal" network as per [EAPTLS] Section 2.1.5.

Further discussion of such provisioning is outside of the scope of this document.

7.5. id-kp-eapOverLAN May not be sufficient

While [RFC4334] Section 2 defines id-kp-eapOverLAN, it gives no explicit use-case for that EKU. That document suggests that the EKU is intended for use in client certificates. However, it also can be read to suggest that the EKU could also be used in server certificates.

As such, we define a new EKU, id-kp-eapClient. If it is determined that this new EKU is not needed, this document can be updated before final publication to use id-kp-eapOverLAN instead of id-kp-eapClient.

7.6. Guest Networks

For EAP-TLS, [EAPTLS] Section 2.1.5 provides for the protocol to be used without peer authentication. The methods outlined here can be extended to perform provisioning within guest networks. That is, the device suspects the identity of the network, but also knows that it does not yet have credentials for use within that network.

Where an EAP peer wishes to connection to the network, but does not know the identity of the network, it SHOULD use EAP-TLS without peer authentication. That is, it should obtain the server certificate without providing a client certificate.

This server certificate can be examined for identification fields, such as id-on-naiRealm. The supplicant SHOULD query the network for the expected server certificate, using the DNS discovery process outlined above.

These certificates and related network configuration which are discovered this way SHOULD NOT be cached for more than one day.

While on the provisioning network, the device can use almost any method to authenticate and authorize the end user. For example, having a "self service" registration page, obtaining temporary credentials, etc.

It is RECOMMENDED that the guest network permit the device to obtain email from anywhere on the Internet, via standard email reception protocols. It is RECOMMENDED that all other ports be blocked. Port 25 (SMTP) MUST be blocked. All DNS ports MUST either be blocked, or be forwarded to a DNS server controlled by the administrator of the guest network.

Network access SHOULD be restricted in both time and usage. There is no reason to allow unauthenticated guest access for more than about 30 minutes. There is no reason to allow unauthenticated guests to transfer gigabytes of data.

These requirements allow the provisioning process to be simple. A device uses EAP-TLS without peer authentication to connect to a network. The user enters an email address in a "self service" registration page. The visited network determines whether or not the person using that email address is authorized. If so, it sends a message to that address with a unique URL.

The user obtains the email, clicks on the URL, and downloads credentials for the visited network. These credentials could be an EAP-TLS client certificate, which has the following properties:

- * issued by the visited network
- * containing information identifying the end user
- * ideally also contains information identifying the device
- * has a limited lifetime, ideally one day
- * lifetime MUST be less than 30 days.

The device then provisions the credentials as above.

This process could also be extended by leveraging DNS even more. The client device could look up a CERT record based on the user's identifying information, e.g. for a user with identifier "user@example.com", visiting a particular naiRealm it could look up a CERT RR of:

```
user.example.com._guest._cert._eap.<naiRealm>
```

The local network could return a custom CERT RR, pointing to a URL for a page which contains a custom client certificate for use with EAP-TLS.

As most DNS servers have limited policy capabilities, this

functionality is likely difficult to implement in practice.

7.7. Using TLS with protocols other than EAP

While the discussion so far has been about EAP, there is no reason to limit this process to EAP. However, we do note that the methods defined here are intended for bootstrapping access to secure networks. They are not intended for use with generic web browsing.

For example, it is possible to use similar methods (though with different DNS names and EKU fields) in order to configure clients for other protocols such as RADIUS/TLS [RFC6614] or RADIUS/DTLS [RFC7360].

These methods are not limited to RADIUS and EAP. For example, discovery of an IMAP server could be done via looking up a SRV record for "_imap._tcp.<naiRealm>", while discovery of an email submission server could be done via looking up "_submit._tcp.<naiRealm>". If a private CA is used for those services, it could be discovered via looking up a CERT record for "_ca._imap._tcp.<naiRealm>".

Similarly, the issue of intermediate CAs discussed earlier is also applicable to other protocols, and therefore requires similar solutions. We also note that there are issues with many other non-WWW protocols which appear to (mis)-use the id-kp-serverAuth field. We offer no solution here to that problem.

It may be possible to use these processes in many other situations, but we do not discuss those use-cases in detail here. We only discuss them as a side note, to demonstrate that automatic provisioning of a client system can be done simply, securely, and with minimal intervention by an end user.

8. Moving to the new methods

The methods given herein are intended to give parties in an EAP conversation more, and better information about what should be happening, and about what is happening. If all of the recommended information is available, then all parties in an EAP conversation have strong, positive indications that the system is secure. If any information is missing or conflicting information is seen, then the system may or may not be secure.

That is, following the recommendations is a positive signal of security. Lack of positive signals does not necessarily indicate insecurity.

It is RECOMMENDED that EAP peers and authenticators which implement these processes add configurable flags which allow the recommendations to be made mandatory. These configurable flags SHOULD permit the recommendations to be enforced in a wide range of conditions, such as per SSID, per realm, per CA, etc. Doing so will allow administrators to make and enforce site-local policies.

For example, a company might mandate that all devices which connect to WiFi use EAP with client certificates, that those client certificates contain the fields defined above, and that those devices only send authentication credentials to EAP authenticators which also satisfy the recommendations above. When an EAP peer follows these mandates, it will not be vulnerable to any of the attacks outlined earlier.

These guidelines allows existing systems to operate unchanged. They also allow updated systems to gain the benefit of increased, and mandated, security.

8.1. Using the new OIDs

In general, we recommend using private CAs for EAP. Such uses avoid the issue of certificate misuse under the [CAB] guidelines.

We also have to address how systems which are unaware of this specification will interact with certificates containing the new OIDs.

Happily, the requirements of [RFC5216] and [EAPTLS] are requirements on what should exist, and not on what should not exist. Tests with implementations, and (where possible) checks of publicly available source code lead us to conclude that these requirements are accurately followed.

The solution then is for server certificates to contain both id-kp-serverAuth, in order to satisfy [RFC5280], and also to contain id-kp-eapServer, in order to satisfy this document. The result is that both old, and new behavior is supported, and that the transition path from one to the other is seamless.

8.2. Recommendations for EAP peers and authenticators

It is RECOMMENDED that EAP peers use a dedicated certificate store for EAP. Where a dedicated certificate store cannot be used, each certificate MUST have additional metadata stored with it, which indicates its permitted uses. This metadata serves as a way of creating "per-use-case" certificate stores.

It is RECOMMENDED that no CAs are enabled by default for EAP. User credentials are provisioned from a known authentication source. If there are no local user credentials configured, then by definition there are no known sources. When credentials are configured, known sources can be enabled at the same time.

It is RECOMMENDED that "web" CAs are not used for EAP. The two use-cases are different, and misuse of certificates opens both EAP and WWW systems to attacks.

It is RECOMMENDED that EAP peers do not perform TLS resumption across different media.

It is RECOMMENDED that when EAP peers use any TLS-based EAP method with a client certificate, that the client certificate contains id-kp-eapClient in order to indicate that the certificate is intended to be used by an EAP peer.

It is RECOMMENDED that EAP peers expose configuration settings which allow the user to permit this new behavior, or require it, on a per-NAI realm basis.

It is RECOMMENDED that EAP servers which permit the use of client certificates mark one or more CAs as being permitted to issue client certificates. These CAs SHOULD be the one which is the "lowest" in the certificate chain. That is, the one which is closest to the client certificate. EAP servers SHOULD NOT mark a global "root" CA as being permitted to issue client certificates, as that root CA may sign many intermediate CAs, each of which could then issue client certificates.

It is RECOMMENDED to use id-pe-wlanSSID [RFC4334] in client and server certificates. When used in a client certificate, it informs the client that this certificate should be used when the given SSID

is seen. When used in a server certificate, it informs the client that the server is intended to be reachable from this particular SSID. Note that a mismatch is not necessarily an error.

It is RECOMMENDED that when EAP authenticators use any TLS-based EAP method with a server certificate, that the server certificate contains `id-kp-eapServer` in order to indicate that the certificate is intended to be used by an EAP authenticator.

It is RECOMMENDED that when EAP authenticators use any TLS-based EAP method with a server certificate, that the server certificate contains one or more `naiRealm`, to indicate that the EAP authenticator is authorized to accept authentication requests for users in those realms.

The requirements of [RFC7585] Section 2.2 on the definition, number, and format of `naiRealm` are included here by reference.

It is RECOMMENDED that when EAP peers use any TLS-based EAP method, that the EAP peer verify that the server certificate presented contains `id-kp-eapServer`, and an `naiRealm` which matches the NAI (if used) in the EAP Identity Response. Any mismatch indicates to the client that the server is not trusted to authenticate users for that realm. Therefore user credentials for that NAI realm should not be sent to the server.

It is RECOMMENDED that when EAP authenticators use any TLS-based EAP method and a client certificate is presented, that the EAP authenticator verify that the client certificate contains `id-kp-eapClient`, and that the NAI (if used) given in the EAP Identity Response field matches one of the `naiRealm` fields in the server certificate. Any mismatch indicates to the server that the client is either misconfigured, or is acting maliciously. The server should therefore treat this mismatch as an authentication failure.

As noted above, the use of the label "*" in `id-on-naiRealm` is forbidden for this specification.

Where the server certificates do not contain `naiRealm`, but do contain one or more `subjectAltName` field of type `dNSName`, clients SHOULD verify that the NAI realm used by the client is an exact suffix of the `dNSName` field.

8.3. Principles and Guidelines

After analysis of the historical practices and standards for EAP, we came to a set of guidelines which are outlined in this section. Application of these guidelines drove the rest of the specification which we define herein.

It is RECOMMENDED that the guidelines given below are followed when developing new network configuration standards and methods:

- * Automated provisioning is strongly preferred to manual provisioning. We define "automated provisioning" as provisioning which is performed via software, with little or no user intervention. Automation minimizes the possibility for end users to create broken or insecure configurations.
- * Manual provisioning should be limited to "Trust on first use" (ToFU), and cached or "pinned" after that. That is, manual provisioning should be limited to allowing a user to approve validation decisions which have been made by the system.
- * Relying on end users to manually configure complex systems is strongly discouraged. Complex systems are difficult to configure, and improperly configured systems create many issues related to security, usability, and network access.
- * Configuration should be "pinned" in order to permit systems to detect and prevent unauthorized changes, and to detect malicious networks which claim to be updated versions of the true network.
- * The identity and role of both parties should be exchanged, and verified. In practice, this suggestion often means that TLS-based EAP methods are preferred to ones which only do name / password credential verification.
- * The previous requirement usually means that the both parties know which RFC 7542 NAI realm is being used. This realm serves a similar purpose to the the DNS host name used in other TLS-based protocols such as HTTPS. As such, similar methods can be used to validate certificate authenticity. This NAI realm is contained in an id-on-naiRealm field, as defined in [RFC7585] Section 2.2
- * For TLS-based EAP methods, trust should be based on a certification authority (CA), which signs certificates for a particular realm. If the CA is trusted, then everything derived from that CA can be trusted. If the CA is not trusted, then it is impossible to trust anything derived from an untrusted CA.

- * CAs should also be associated with permitted uses. For example, a root CA which is trusted for web surfing is not necessarily trusted for use with EAP authentication. In practice this means either having separate certificate stores for different purposes, or annotating root certificates with their permitted uses.

We believe that these recommendations are correct, simple, practical, and will improve security and usability for all participants in EAP. We show that there is a clear upgrade path from current behavior to better behavior. Each step of that upgrade path is simple, and involves minimal change for end users or administrators.

9. Security Considerations

There are a large number of security issues with current practices. This document attempts to give both fixes, and a transition path to a better system. As such, the entire document is discussing security issues.

One of the main points of this document is that systems which are difficult to configure are likely to be insecure.

This document also highlights problems with misuse of certificates containing id-kp-serverAuth, and id-kp-clientAuth. If such misused certificates were to be widely reported, then large parts of the Internet could be taken offline.

We note that distribution of these certificates MUST NOT be done via email. There is just too much possibility for forgery and user mistakes for that process to be secure. We instead rely on secure transport layers and cryptographically signed data in order to bootstrap authenticated network access.

9.1. On Identities and Service Discovery

All the user needs is an identity (e.g. "user@example.com"), and a password. Essentially everything else required for network access can be derived automatically, and provisioned with no additional user input. This process is significantly more secure than manual provisioning.

9.2. Password Hashing and Storage

In some situations, using client certificates with EAP is preferable to using password-based methods. Password-based EAP methods often hash the password along with a salt or a challenge, and then send the hashed version of the password. However, this hashing can conflict with the desire of administrators to store hashed passwords in their

user databases. The two different hashing methods are almost always incompatible, which means that the administrator has to choose either to send passwords via a method such as PAP inside of TTLS, or to store clear-text passwords in their local user database.

Neither choice is optimal. Where there is a trade-off, it is RECOMMENDED that systems use a method such as TTLS with PAP, and then store hashed or encrypted passwords in the local user database. The "clear-text" password which is sent in TTLS is, in fact, secured via TLS when it is sent "over the wire". So it is incorrect to claim that EAP is sending passwords "in the clear".

The caveat here is that supplicants must verify the identity of the server certificate before sending the password to the EAP server. As we have seen in Section 2 above, many supplicants skipped this step. As a result, it has been common practice to recommend that supplicants use EAP methods which do not send clear-text passwords. However, all this recommendation does is to move the security risk from the supplicant to the database, which is now required to store clear-text passwords for all users.

While some would argue that exposing the users clear-text password to an EAP Server is a security risk, it is in practice irrelevant. The EAP server is almost always co-located with an AAA server (e.g. RADIUS or Diameter). Those servers control network access for entire organizations, including setting complex policies. Any attacker who gains control of an AAA server can take many more, and worse actions than to simply observe peoples passwords.

In contrast, history shows that exposure of user databases (with names and passwords) is not uncommon. In fact, as the EAP or AAA server usually has complete access to the user database (including passwords), compromise of the AAA server almost by definition leads to compromise of the local user password database.

We therefore make the trade-off which has the lowest possible security impact, for all failure cases. Passwords SHOULD be stored hashed or encrypted in a user database. TLS-based EAP methods which rely on passwords SHOULD use authentication methods which are compatible with such password storage methods, which generally means that the passwords are sent by the user in clear-text, but are protected by TLS.

10. IANA Considerations

This section this specification requests from Internet Assigned Numbers Authority (IANA) registration of the following items.

10.1. Key Purpose OIDs

We request registration of values related to the certificate key purpose OIDs in accordance with [RFC8126].

* id-kp-eapServer

* id-kp-eapClient

10.2. Underscored and Globally Scoped DNS Node Names

Per RFC 8552, please add the following entry to the "Underscored and Globally Scoped DNS Node Names" registry:

RR Type	_NODE NAME	Reference
CERT	_ca._cert._eap	<this document>
CERT	_client._cert._eap	<this document>
CERT	_server._cert._eap	<this document>
SRV	_est._eap	<this document>
URI	_install._mdm	<this document>

We note that [RFC8552] does not provide for "sub" registries, as we have defined above. However, we believe that these definitions fall within both the intent of [RFC8552], and common practice.

11. References

11.1. Normative References

[RFC2119]

Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", RFC 2119, March, 1997, <<http://www.rfc-editor.org/info/rfc2119>>.

[RFC3748]

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

[RFC4334]

Housley, R., and Moore, T., "Certificate Extensions and Attributes Supporting Authentication in Point-to-Point Protocol (PPP) and Wireless Local Area Networks (WLAN)", RFC 4334, February 2006

- [RFC4398]
Josefsson, S., "Storing Certificates in the Domain Name System (DNS)", RFC 4398, March 2006.
- [RFC5216]
Simon, D., Aboba, B., and R. Hurst, "The EAP-TLS Authentication Protocol", RFC 5216, March 2008
- [RFC7170]
Zhou, H., et al., "Tunnel Extensible Authentication Protocol (TEAP) Version 1", RFC 7170, May 2014.
- [RFC7234]
Fielding, Ed., et al, "Hypertext Transfer Protocol (HTTP/1.1): Caching", RFC 7234, June 2014
- [RFC7542]
DeKok, A., "The Network Access Identifier", RFC 7542, May 2015.
- [RFC8126]
Cotton, M., et al, "Guidelines for Writing an IANA Considerations Section in RFCs", RC 8126, June 2017.
- [RFC8174]
Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", RFC 8174, May 2017, <<http://www.rfc-editor.org/info/rfc8174>>.
- [RFC8552]
Crocker, D., "Scoped Interpretation of DNS Resource Records through "Underscored" Naming of Attribute Leaves", RFC 8552, March 2019.
- [EAPTLS]
Mattsson, J., and Sethi, M., "Using EAP-TLS with TLS 1.3", draft-ietf-emu-eap-tls13-15, May 2021.

11.2. Informative References

- [CAT]
<https://cat.eduroam.org/>
- [EDUROAM]
<https://eduroam.org/>
- [MSPEAP]
<https://msdn.microsoft.com/en-us/library/cc238354.aspx>

[PEAP]

Palekar, A. et al, "Protected EAP Protocol (PEAP)", draft-josefsson-pppext-eap-tls-eap-06.txt, March 2003.

[CAB]

CA/Browser Forum, "Baseline Requirements for the Issuance and Management of Publicly-Trusted Certificates" Version 1.7.4, 5 April 2021 <https://cabforum.org/wp-content/uploads/CA-Browser-Forum-BR-1.7.4.pdf>

[RFC2716]

Aboba, B., and Simon, D., "PPP EAP TLS Authentication Protocol", RFC 2716, October 1999.

[RFC2865]

Rigney, C., Willens, S., Rubens, A., and W. Simpson, "Remote Authentication Dial In User Service (RADIUS)", RFC 2865, June 2000.

[RFC2818]

Rescorla, E., "HTTP Over TLS", RFC 2818, May 2000.

[RFC4851]

Cam-Winget, N., et al, "The Flexible Authentication via Secure Tunneling Extensible Authentication Protocol Method (EAP-FAST)", RFC 4851, May 2007.

[RFC5280]

Cooper, D., et al., "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, May 2008.

[RFC5281]

Funk, P., and Blake-Wilson, S., "Extensible Authentication Protocol Tunneled Transport Layer Security Authenticated Protocol Version 0 (EAP-TTLSv0)", RFC 5281, August 2008.

[RFC5785]

Nottingham, M. and Hammer-Lahav, E., "Defining Well-Known Uniform Resource Identifiers (URIs)", April 2010.

[RFC5931]

Harkins, D., and Zorn, G., "Extensible Authentication Protocol (EAP) Authentication Using Only a Password", RFC 5931, August 2010.

[RFC6614]

Winter, S., et al., "Transport Layer Security (TLS) Encryption for RADIUS", RFC 6614, May 2012.

[RFC6677]

Hartman, S. (ed), et al., "Channel-Binding Support for Extensible Authentication Protocol (EAP) Methods", RFC 6677, July 2012.

[RFC7030],

Pritikin, M. (Ed), Et al, "Enrollment over Secure Transport", October 2013

[RFC7360]

DeKok, A., "Datagram Transport Layer Security (DTLS) as a Transport Layer for RADIUS", RFC 7360, September 2014.

[RFC7585]

Winter, S, and McCauley, M., "Dynamic Peer Discovery for RADIUS/TLS and RADIUS/DTLS Based on the Network Access Identifier (NAI)", RFC 7585, October 2015.

[RFC7593]

Weiringa, K. et al, "The eduroam Architecture for Network Roaming", RFC 7593, September 2015.

[RFC8446]

Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", August 2018.

[RFC8894]

Gutmann, P., "Simple Certificate Enrolment Protocol", RFC 8894, September 2020.

Acknowledgments

This document would not be possible without the input of many people.

Jorge Vergara provided reviews of previous documents which led to a clearer articulation of the problem described here, and which therefore motivated this document. He has also provided reviews for early versions of this document.

Tom Rixom provided a significant amount of information on issues seen by supplicants, which motivated much of the text in Section 3.1.

Arran Cudbard-Bell suggested using DNS for the "out of band" provisioning of certificates in Section 3. He also provided detailed reviews of multiple versions of this document.

Stefan Winter provided feedback on a number of issues related to certificates, roaming, and provisioning.

Karri Huhtanen raised issues related to purpose-specific CAs, and provided suggestions to help address those issues.

Authors' Addresses

Alan DeKok
Network RADIUS SAS
26 rue Colonel Dumont
38000 Grenoble
FRANCE

Email: aland@networkradius.com

Independent Submission
Internet-Draft
Intended status: Informational
Expires: December 12, 2021

K. Makhijani
L. Dong
Futurewei
June 10, 2021

Requirements and Scenarios for Industry Internet Addressing
draft-km-industrial-internet-requirements-00

Abstract

Industry Control Networks host a diverse set of non-internet protocols for different purposes. Even though they operate in a controlled environment, one end of industrial control applications run over internet technologies (IT) and another over operational technology (OT) protocols. This memo discusses the challenges and requirements relating to convergence of OT and IT networks. One particular problem in convergence is figuring out reachability between these networks.

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 12, 2021.

Copyright Notice

Copyright (c) 2021 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	2
2. Terminology	3
2.1. Acronymns	4
3. Industrial Network Reference Architecture	4
3.1. Communication Patterns	5
3.2. Industry Control Network Nuances (current state)	5
4. Problem Statement	6
4.1. Heterogenity	7
4.2. Automation Impact	7
4.2.1. Scale	8
4.2.2. Stretch Control Fabric to Edge and Cloud	8
4.2.3. Reliability	8
4.2.4. Resilience	8
4.3. OT/IT Convergence	8
4.4. Data oriented networking	9
4.5. Virtualization	9
5. Address Space Requirements	9
5.1. Short Device Addressing	9
5.2. Meaningful Addresses	10
5.3. Device name based Addresses	10
5.4. Adoption of Lean Network Layer	10
5.5. Multi-semantic behavior	10
5.6. Interoperability with IP-world machines	11
6. Relationship with Activities in IETF	11
6.1. Deterministic Networks (DetNet WG)	11
6.2. IoT OPS	11
6.3. LPWAN	11
6.4. Recent Addressing related work	12
7. IANA Considerations	12
8. Security Considerations	12
9. Acknowledgements	12
10. Informative References	12
Authors' Addresses	13

1. Introduction

An industry control network interconnects devices used to operate, control and monitor physical equipment in industrial environments. These networks are increasingly becoming complex as the emphasis on convergence of OT/IT grows to improve the automation. On one side of Industrial internet are the inventory management, supply chain and simulation software and the other side are the control devices

operating on machines. Operational Technologies (OT) networks are more often tied to set of non-internet protocols such as Modbus, Profibus, CANbus, Profinet, etc. There are more than 100 different protocols each with it's own packet format and are used in the industry.

It is expected that integration between the IT and OT will provide numerous benefits in terms of improved productivity, efficiency of operations by providing end to end visibility and control. Industry control applications also expect to operate at cloud scale by virtualization of several modules (especially PLCs) leading to new set of network requirements.

One aspect of industry control is the delivery of data associated with the Real-time, deterministic and reliability characteristics over local-area and wide-area networks. This type of inter-operability functionality and study is already covered in DETNET working group. The other aspect is reachability and interconnection keeping heterogeneity of communication interfaces and a variety of services in mind. This document focuses on the latter part only.

OT networks have been traditionally separate from the IT networks. It allowed OT network experts to manage and control processes without much dependency on changes in the external networks. This is an important to consideration when dealing with the industry control networks to maintain them in a controlled environment leveraging the limited-domain networks [LDN] concept for an independent network control.

The purpose of this document is to discuss the reachability and interconnection characteristics, challenges and new requirements emerging from large-scale integration of IT and OT.

2. Terminology

- o Industrial Control Networks: The industrial control networks are interconnection of equipments used for the operation, control or monitoring of machines in the industry environment. It involves different level of communications - between fieldbus devices, digital controllers and software applications
- o Industry Automation: Mechanisms that enable machine to machine communication by use of technologies that enable automatic control and operation of industrial devices and processes leading to minimizing human intervention.
- o Human Machine Interface: An interface between the operator and the machine. The communication interface relays I/O data back and

forth between an operator's terminal and HMI software to control and monitor equipment.

2.1. Acronyms

- o HMI: Human Machine Interface

3. Industrial Network Reference Architecture

In the scope of this document the following reference industrial network will be used to provide structure to the discussion. In the Fig. Figure 1 below, a hierarchy of communications is shown. At the lowest level, PLCs operate and control field devices; above that Human Machine Interface (HMI) interconnects with different PLCs to program and control underlying field devices. HMI itself, sends data up to applications for consumption in that industry vertical.

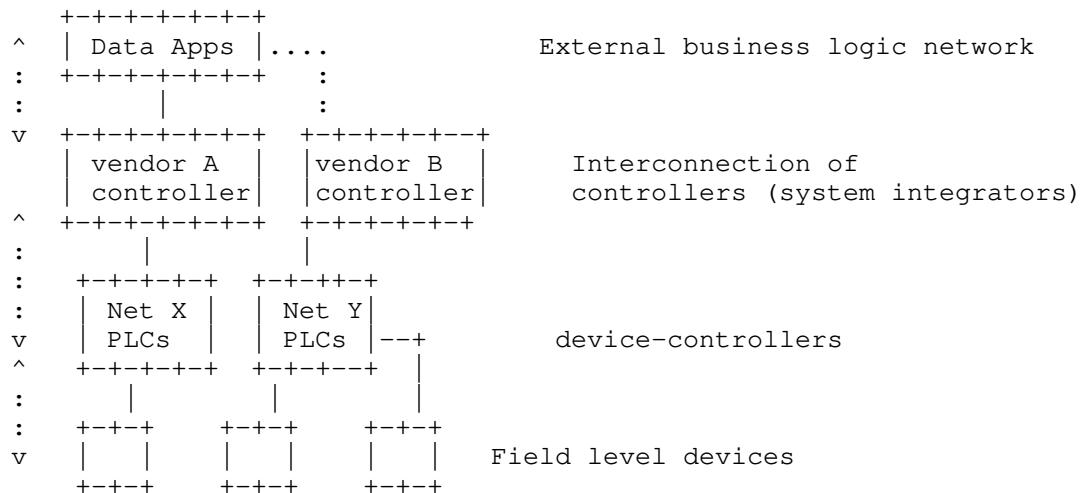


Figure 1: Hierarchy of Functions Industrial Control Networks

Unlike commercial networks that uniformly run IP protocols, the communication links run different protocols at along the different level of the hierarchy. One of the key requirement from new industrial applications is the integration of different types of communication protocols including Modbus, Profinet, Profibus, ControlNet, CANOpen etc.

A vertically integration system involves a network between the external business applications and higher controllers (for e.g. SCADA, HMI, or system integrators) is IP based. The second level of networks between the controllers can be either IP or non-IP

(Profibus, BACNet, etc.). The lowest field-level networks between industrial controllers and field-level may be any of the fieldbus or device control protocols (More details of the industry networks can be found in [SURV]).

3.1. Communication Patterns

The following communication patterns are commonly observed:

- o controller to controller: A communication between multi-vendor controller maybe required by system integrators to work in complex systems.
- o controller to field level devices: This is a fieldbus communication between device such as I/O modules, motors, controllers. This communication represent.
- o Device to device: allows direct communication between wired industrial devices and wireless devices to enhance automation use cases. For an exmaple, use of camera to visually monitor and detect anamolies in other devices.
- o controller to compute: vertical communication between a controller and compute integrates IP-based technologies with non-IP since OT product systems and solutions are not connected with IP based networks.

A certain level of inter-operability is required to exchange data between the above endpoints from different vendors. One of the challange is that Ethernet (which unifies IT standards) that's not always possible in Industry networks.

3.2. Industry Control Network Nuances (current state)

The Industry control networks are engineered for the idustry verticals they belong to and depict unique properties as below:

- o location bound: The Control Device's location or the network they are attached to is predetermined and changes rarely. However, the network resources may not get efficiently utilized to avoid contention between them.
- o security by separation: Typically, security is enhanced by keeping IT/OTnetworks separate. The operators control how data goes in and out of a site through firewalls and policies.
- o data growth: Even though the size of network remains the same, data generated is much higher. For example, cameras installed for

visual inspection to determine the quality of manufactured product generates a high bandwidth demand.

- o Wired device constraints: A bulk of machines are over wired network, their constraints vary from LPWAN and IoT devices which is an active area of standardization work. device lifetime, or power-requirements are not typical constraints. Instead direct process control mechanisms are more important.
- o Real-time behavior: The control devices require realtime as well as deterministic behavior between a controller (such as an HMI station) to the equipment. The DetNet working group covers several aspects.

The goal of the document is not to reinvent the Industry control infrastructure. See section Section 6 on related standards work. It is meant to exclude the already covered by other WGs.

Since a device connects to network through its address, the document explores different address specific nuances in control devices - such as management, device discovery and integration requirements. This document concerns with the identification of and role networks, specifically from the organization of industry control devices.

The goal of this document is to outline some of the challenges and improvement of connectivity aspects of Industry control networks.

4. Problem Statement

In industrial networks, a good number of devices still communicate over a serial or field bus (although Ethernet is being gradually adopted). The operations on these devices are performed by writing provide direct access to operation-control. i.e what operation to perform is embedded in the type of interface itself. For instance, Profibus, Modbus networks are implicitly latency sensitive and short control-command based.

ModBus

```

+---+---+---+---+---+---+---+---+---+---+
| address | Function  code  | data |
+---+---+---+---+---+---+---+---+---+---+

```

CANBus

```

+---+---+---+---+---+---+---+---+---+---+
| message id | data |
+---+---+---+---+---+---+---+---+---+---+

```

Profibus - todo.

Since they are localized in an area such as factory floor or a site, such networks have evolved independently and are separated from the IT applications. The emerging trend requires a seamless integration with intelligent software, sophisticated compute platforms and other operational aspects as highlighted below:

4.1. Heterogeneity

A typical industry control network has devices of different communication interfaces such as Fieldbus (PROFIBUS, Modbus, and HART), Ethernet (generic Ethernet/IP, PROFINET, and Modbus-TCP), and also wireless (Bluetooth, Wireless HART, and IoT). These interfaces vary at the physical and link layers and because they integrate with their own application technologies providing interoperability between these devices remains a challenge. This also makes difficult to adopt to modern integration technologies.

Fieldbus client-server architecture is widely deployed. It delivers commands deterministically from a controller to the device and vice-versa. Interfaces of this kind have typically shorter addresses (upto 256 devices on a single bus in Modbus).

Some of the servers also behave as protocol gateways and interconnect different type of protocols. For example when a modbus device is being controlled by a profinet server, an gateway function will translate modbus data or encapsulate it over IP (if the controller supports it).

In a Gateway-centric approach, gateways are in charge of protocol translations between the devices with different interfaces. This requires packing and unpacking of data in the source and destination formats at the attached gateways. Note: As an example, a Modbus device does not know whether to send command to Profibus PLC or Modbus PLC. The gateway device attaches to performs the translation. This is even worse with encapsulations, where the entire frame is carried over IP.

This is not ideal for latency sensitive applications. Although hardware wise, gateways need to be equipped with all the interface, it is more efficient to only perform data link conversion.

4.2. Automation Impact

Automation of processes in industry relies on control sophisticated technologies such as machine learning, big data, etc. with minimal human intervention. Automation needs to support scale, reliability and resilience at large-scale.

4.2.1. Scale

Automation control at small scale applications with well defined task has been possible. In order to improve production, and eliminate stoppages and minimizing human intervention.

When the number or density of devices, and processes increase there is a need to schedule, route, and coordinate over multiple control environments.

4.2.2. Stretch Control Fabric to Edge and Cloud

The industry control networks can be extended to cloud or edge compute platforms. Since these networks are not equipped with compute intensive servers. Now extending the communication to the edge and cloud nodes increases the distance requiring traditional L2 networks to be adopted to L3 network designs.

Design decisions will require to choose different transit strategies (this maybe layer 1, 2, 3 technologies or even network slices). It also influence the security policies.

4.2.3. Reliability

Production efficiency is inversely related to number of defects in a process. System reliability is determined through measurements of its instantaneous state.

Automation processes need to ensure that system is performing in an expected state and is capable of reporting anomalies fast and accurately (i.e. packet drops or jitter leading to poor quality product).

4.2.4. Resilience

TBD.

4.3. OT/IT Convergence

Most of the factory floors are not equipped with IT servers to perform compute intensive tasks. Yet an IP-based device need to connect with non-IP interface to control those devices.

Often real-time response is necessary for example, in closed-loop control systems direct communication is desired to avoid any additional packet processing delay or overheads at the source and destination gateways, equipping IP to all OT devices and abandoning

the existing investment and depolyment could result in the following obvious problems.

- o Many of the standard IP based protocols maybe too much overhead for OT devices.
- o Cannot preserve communication characteristics of devices (different device addressing scheme, realtime, IRT, message identifiers, Bus-like properties).
- o It relies heavily on hierarchy network stack (network layer, transport layer, application), where as OT devices do not have any, they generally operate at data link layer or below.

4.4. Data oriented networking

Industry verticals keep data and control on the manufacturing floor, on a closed system. There is no easy way to forward this data to enterprise level software. On premise micro data centers or edge computing are new infrastructure pieces that will impact the design of current industrial networks.

4.5. Virtualization

Traditional Industry control infrastructure is not virtualized. Virtualization will enable deployment of new functionality in a flexible manner.

- o Virtual PLCs are considered an important component functionality customization of digital-twin realization.
- o virtualization enables edge and cloud native computing by moving and instantiating workflows at different locations.

Implications that PLCs are no longer one-hop away.

5. Address Space Requirements

5.1. Short Device Addressing

Shorter addresses are inherent to industry control systems to provide implicit determinism.

Note: The motivation for short address is to preseve the legacy attributes of fieldbus control devices. It is not related low-power or resource constraints.

A large volume of the messages are of sizes shorter than the size of IP headers (v4, v6) themselves. The header tax will be very high over industry control networks.

5.2. Meaningful Addresses

The industry control floors are built bottom-up. The devices are carefully wired and connected to controllers. In a hierarchical network design, a particular type of machine can be reached in a structured manner by adding subnet or location to the address structures.

5.3. Device name based Addresses

HMI might require human readable address that is understandable to human operators or application end users. For example, a device address could be associated with its location, type of applications, attached objects etc. The network needs to support the resolution and routing based on such device addresses, which is more user friendly. On the other hand, grouping devices based on their addresses shall be easily implemented to enable group operation and communication.

5.4. Adoption of Lean Network Layer

Challenge of Industrial network device address is that it communicates to a physical device address. Traditionally, in a limited environment there was no need for network layer or expressing network specific service, access control.

- o If a sensor is broken, it will require reprogramming of controller and re-aligning with the new address. The benefit of network layer, removes this restriction.
- o Note that, using IP stack is not suitable because these devices perform specific functions and any overhead in transport or large addressing can add to processing delays.
- o Several other IP suite protocols such as device discovery should be revisited.

5.5. Multi-semantic behavior

OT networks, at least at site level are organized at much smaller scale than typical IP-capable networks. There is in turn a fixed hierarchy of networks w.r.t. location in a plant.

5.6. Interoperability with IP-world machines

To develop further on different type of address format support. From smaller address of legacy devices to IT based applications with IP address.

(OT-Address)--->(Industry Control)--->(IP-Address)
(control dev) (network) (application)

Preferably allow OT devices to understand IP-addresses for the servers they connect to.

6. Relationship with Activities in IETF

6.1. Deterministic Networks (DetNet WG)

The Deterministic Networking (DetNet) [DETNET-ARCH] is working on using IP for long-range connectivity with bounded latency in industry control networks . Its data plane [DETNET-DP] takes care of forwarding aspects and most close to Industry control networks but the focus is on the controlled latency, low packet loss & delay variation, and high reliability functions. Not dealing with interconnection of devices.

In layer 2 domain, similar functionality is covered by TSN Ethernet [IEEE802.1TSNTG].

6.2. IoT OPS

IoT operations group discusses device security, privacy, and bootstrapping and device onboarding concepts. Among the device provisioning one of the object is network identifier. We understand that the IoT OPS does not exclude evaluation of industry IoT or control devices requirements. Given the specific functions described above it maybe necessary to configure more than an identifier, i.e. server or controller information or specific address scope and structure.

6.3. LPWAN

The LPWAN has focussed on low-power and constrained devices. There are compression related approaches that may apply are [SCHC] or [ROHC]. To be evaluated for process control devices.

6.4. Recent Addressing related work

Some of the work initiated on the addressing include solutions such as [FlexIP], [Flexible_IP], [FHE], and [SOIP].

Recently, a broader area of problem statement and challenges in [CHALLENGE].

7. IANA Considerations

This document requires no actions from IANA.

8. Security Considerations

This document introduces no new security issues.

9. Acknowledgements

10. Informative References

[CHALLENGE] Jia, Y., Trossen, D., Iannone, L., 3rd, D. E. E., and P. Liu, "Challenging Scenarios and Problems in Internet Addressing", draft-jia-intarea-scenarios-problems-addressing-00 (work in progress), February 2021.

[DETNET-ARCH]
Finn, N., Thubert, P., Varga, B., and J. Farkas,
"Deterministic Networking Architecture", RFC 8655,
DOI 10.17487/RFC8655, October 2019,
<<https://www.rfc-editor.org/info/rfc8655>>.

[DETNET-DP]
Varga, B., Ed., Farkas, J., Berger, L., Fedyk, D., and S. Bryant, "Deterministic Networking (DetNet) Data Plane: IP", RFC 8939, DOI 10.17487/RFC8939, November 2020,
<<https://www.rfc-editor.org/info/rfc8939>>.

[FHE] Jiang, S., Li, G., and B. Carpenter, "Asymmetric IPv6 for Resource-constrained IoT Networks", draft-jiang-asymmetric-ipv6-04 (work in progress), November 2020.

[Flexible_IP]
Jia, Y., Chen, Z., and S. Jiang, "Flexible IP: An Adaptable IP Address Structure", draft-jia-flex-ip-address-structure-00 (work in progress), October 2020.

- [FlexIP] Moskowitz, R., Li, G., and S. Ren, "FlexIP Addressing", draft-moskowitz-flexip-addressing-00 (work in progress), January 2019.
- [IEEE802.1TSNTG] "IEEE, "Time-Sensitive Networking (TSN) Task Group", 2018, <<https://1.ieee802.org/tsn>>.
- [LDN] Carpenter, B. and B. Liu, "Limited Domains and Internet Protocols", RFC 8799, DOI 10.17487/RFC8799, July 2020, <<https://www.rfc-editor.org/info/rfc8799>>.
- [ROHC] Jonsson, L-E., Pelletier, G., and K. Sandlund, "The RObust Header Compression (ROHC) Framework", RFC 4995, DOI 10.17487/RFC4995, July 2007, <<https://www.rfc-editor.org/info/rfc4995>>.
- [SCHC] Minaburo, A., Toutain, L., Gomez, C., Barthel, D., and JC. Zuniga, "SCHC: Generic Framework for Static Context Header Compression and Fragmentation", RFC 8724, DOI 10.17487/RFC8724, April 2020, <<https://www.rfc-editor.org/info/rfc8724>>.
- [SOIP] Carpenter, B., Jiang, S., and G. Li, "Service Oriented Internet Protocol", draft-jiang-service-oriented-ip-03 (work in progress), May 2020.
- [SURV] Galloway, B. and G. Hancke, "Introduction to Industrial Control Networks", IEEE Communications Surveys & Tutorials Vol. 15, pp. 860-880, DOI 10.1109/surv.2012.071812.00124, 2013.

Authors' Addresses

Kiran Makhijani
Futurewei

Email: kiran.ietf@gmail.com

Lijun Dong
Futurewei
Central Expy
Santa Clara, CA 95050
United States of America

Email: lijun.dong@futurewei.com

IOTOPS
Internet-Draft
Intended status: Informational
Expires: January 13, 2022

B. Moran
Arm Limited
July 12, 2021

A summary of security-enabling technologies for IoT devices
draft-moran-iot-nets-00

Abstract

The IETF regularly develops new technologies. Sometimes there are several standards that can be combined to become vastly more than the sum of their parts. Right now, there are six technologies either recently adopted or poised for adoption that create such a cluster. Combining secure onboarding, remote attestation, secure update, software bill-of-materials/expected attestation, automated network policy enforcement, and trusted execution environment provisioning, devices can be defended from many threats. This is an opportunity for an inflection point for more secure and trustworthy devices. Simultaneous adoption of two or more of these six standards could create the foundation of computing devices that are worth trusting.

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 January 13, 2022.

Copyright Notice

Copyright (c) 2021 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	2
2. Barriers to IoT Adoption	3
3. Foundations of Trustworthy IoT	3
3.1. Detecting a Compromise	4
3.2. Halting Malicious Activity	5
3.3. Remedying Vulnerabilities	5
4. Baseline Requirements for Secure Networks	5
5. IoT Technologies for Secure Networks	6
5.1. Trust Relationships in Secure IoT Networks	7
6. Normative References	8
Author's Address	9

1. Introduction

IoT devices (unattended devices with network connections) are often considered a weak point in networks and have often been used by malicious parties to extract information, serve as relays, or mount attacks. Appropriate use of security technologies can mitigate this trend and enable users allow for security policies that do not have to be overly protective of IoT systems and enable them to add the full potential of value they were designed to add.

This draft addresses six trustworthiness problems in IoT devices and proposes solutions to them with six technologies. The problems are:

1. What software is my device running?
2. How should my device connect to a network?
3. With which systems should my device communicate?
4. What is the provenance of my device's software?
5. Who is authorised to initiate a software update and under what circumstances?
6. How should my device update its trusted software?

Each of these questions is answered by recently developed or developing standards.

2. Barriers to IoT Adoption

IoT adoption is generally presented as a platform problem or a data acquisition and analysis problem. The result is a proliferation of communication formats, radio standards, network technologies, operating systems, data gathering schemes, etc. Despite this effort, IoT is not growing at the projected rates.

IoT is not simply a combination of a device platform and a data-gathering platform. In the life-cycle of devices, they must be commissioned and onboarded. When a flaw is discovered, they must be updated to restore trustworthiness and there must be evidence that they are effectively trustworthy (e.g. running the intended/expected software). Acknowledging the chance of security breaches, network infrastructure must be configured to allow access to necessary services and restrict access to everything else.

Commissioning, onboarding, attestation, update, and access control are complex core technologies that are difficult to implement well. This can be seen with the plethora of poorly implemented IoT devices that have been reported in the news whenever a defect is found.

IoT adoption is hampered by a lack of core technologies surrounding the development of trustworthy devices and device trustworthiness. These core technologies do not present obvious revenue streams and they require cooperation between many vendors for them to succeed, which may explain the low rate of innovation in this space.

To reduce this barrier to entry, the IETF has been investing in these core technologies.

3. Foundations of Trustworthy IoT

IoT devices can bring a lot of value to businesses and individuals, but they are also difficult to manage because of their diversity, difficulty in auditing, maintenance, onboarding practices, and lack of visibility about device security posture and device software.

Initiatives such as PSA Certified focus on device level security principles and encourage the use of a hardware Root of Trust (RoT) that provides a source of confidentiality and integrity for IoT systems. The security principles and security requirements of PSA Certified Level 1 cover topics such as trusted boot, validating updates, attestation and secure communications. Complementary to this, IETF provides standards that can be used to create secure

networks; this memo focuses on six standards that can beneficially be used together at the network level.

Building trustworthy IoT is about more than just building devices conforming to best-practice security. Users, Owners, Operators, and Vendors must be able to respond when a compromise occurs. Responding to compromised IoT consists of three key points:

1. Detecting a compromise
2. Halting malicious activity
3. Remedying vulnerabilities and flawed software.

Once a compromise has been detected, the affected device needs to be quarantined from the network, then security patches must be applied.

3.1. Detecting a Compromise

There are two broadly applicable ways to remotely detect a compromised IoT device:

1. Detect anomalous software on the device.
2. Detect anomalous network traffic from the device.

Detecting anomalous software on the device requires remote attestation of software measurements; the report of what software the device is running must be trustworthy even if the software is not. However, attestation is an incomplete solution if the recipient of attestation evidence does not know what to expect. Hence, trustworthy and authoritative sources with understanding of what is to be expected are required. Furthermore, automated systems for delivering these expected values must be very secure or else they will become targets for threat actors as well.

Detecting anomalous traffic from the device requires a baseline of expected traffic; otherwise, network infrastructure cannot know what traffic is legitimate and what is not. This expected traffic information needs to be closely associated with each individual device, since network traffic patterns may shift from device to device or version to version. These trustworthy and authoritative sources of patterns must also be protected: a compromised device could report an incorrect expected network traffic pattern, or a threat actor could modify an expected network traffic pattern.

3.2. Halting Malicious Activity

Halting malicious activity is done by network infrastructure. A Network Access Control (NAC) system, such as a router, gateway, firewall, or L3 managed switch, can apply a network access policy on a per-device basis. The NAC system uses a policy that is provided to it in advance in order to determine the access requirements of each connected autonomous device. Assuming that these policies are constructed according to the principle of least privilege, This allows the NAC system to drop any communication that does not match the defined policies, effectively eliminating the use of IoT devices as relays, proxies, or mechanism to pivot in a network. It may even prevent compromises before they occur because inbound traffic to IoT devices that does not conform to policy can be discarded.

For shared media, such as radio protocols, intra-LAN policies cannot be preemptively effectively enforced, but they can be monitored and enforced after violation, for example by removing network access rights. Per-device Internet-to-LAN policies and LAN-to-Internet policies can still be applied as normal.

3.3. Remedying Vulnerabilities

Remedying vulnerabilities requires a remote update system. Where there are secure components that are independently updatable, additional considerations are required. In both cases, the new software must be signed, but that alone is insufficient: new software must be authenticated against a known, authorized party. It must also come with a statement of provenance: a software bill of materials or SBOM. This statement must describe all the components of the software along with defining the authorship of the software, which may be separate from the authority to install that software on a given device.

4. Baseline Requirements for Secure Networks

To establish a trustworthy IoT network, devices MUST be able to prove:

1. What software they are running and, by extension:
 1. The provenance of the software.
 2. (Optionally) that it has been checked for common malware, backdoors, etc.
2. Who they will connect to or exchange data with so that anomalies can be registered.

To install and maintain IoT devices, authorized entities MUST be able to:

1. Connect a device to a secure network.
2. Initiate an update of a device.
3. (Optionally) Add or remove authorized entities from the device.
4. (Optionally) Deploy and remove protected assets to and from the device.

Each of these requirements stops a particular avenue of attack against device, networks, or data collection systems.

5. IoT Technologies for Secure Networks

Assembling the foundations of trustworthy IoT and the baseline requirements for secure networks, the result is a set of requirements, described here with enabling standards:

1. To deploy new keys into a device and connect it to a network, devices SHOULD support a secure onboarding protocol such as FIDO Device Onboarding [FDO] or LwM2M Bootstrap ([LwM2M]).
2. To enable devices to report their current software version and related data securely, devices SHOULD support a support a mechanism of performing attestation measurements in a trustworthy way and a Remote Attestation protocol, such as [I-D.ietf-rats-eat].
3. To enable devices to be updated securely in the field, they SHOULD support a remote update protocol such as [I-D.ietf-suit-manifest].
4. To prove the provenance of a firmware update, update manifests SHOULD include (directly, or by secure reference) a Software Identifier or Software Bill of Materials, such as [I-D.ietf-sacm-coswid].
5. To enable a Relying Party of the Remote Attestation to correctly evaluate the Attestation Report, the SBoM (such as [I-D.ietf-sacm-coswid]) SHOULD contain expected values for the Attestation Report.
6. To ensure that network infrastructure is configured discern the difference between authentic traffic and anomalous traffic, network infrastructure SHOULD contain a [RFC8520] Manufacturer

Usage Description (MUD) Controller which accepts MUD files in order to automatically program rules into the network infrastructure.

7. In order for network infrastructure to be configured in advance of any changes to devices, MUD files SHOULD be transported (directly or by secure reference) within update manifests.
8. To enable rapid response to evolving threats, the MUD controller SHOULD also support dynamic update of MUD files.
9. Network infrastructure SHOULD apply risk management policy to devices that attest non-compliant configuration. For example, a device with out-of-date firmware may only be permitted to access the update system.

5.1. Trust Relationships in Secure IoT Networks

[FDO] and [LwM2M] enable the installation of trust anchors in IoT devices. These enable the services to ascertain that the devices are not counterfeit. They also enable the devices to trust that the services are not on-path attackers.

The combination of SUIIT, CoSWID and RATS Attestation secures these trust relationships further. A device operator receives a SUIIT manifest, that contains a CoSWID. They apply the SUIIT manifest to a device. The newly updated device then attests its software version (one or more digests) to the device operator's infrastructure. The device operator can then automatically compare the attestation evidence to the CoSWID.

The device operator can trust that expected values are correct because they are signed by the software author. The device operator can trust that the attestation report is correct because it is signed by the verifier and, finally, the device operator can trust the device because its attestation evidence content matches its CoSWID.

To extend this relationship to Trusted Applications (TAs) as well, devices that support TAs can also implement [I-D.ietf-teep-architecture].

Adding MUD to the combination above cements the established trust with enforcement. The network operator also receives the SUIIT manifest for the device. The manifest contains a MUD file in addition to the above. The device does not need to report a MUD URI as described in [RFC8520]-which stops the device from falsifying it. Instead, the network operator also receives an attestation report for the device. If the attestation report matches the CoSWID in the

manifest, then the network operator automatically applies the MUD file that is also contained in the manifest. This allows a secure link to be established between a particular MUD file and a particular software version.

The trust relationships are somewhat more complex with MUD: the network operator may not trust the software author to produce vulnerability-free software. This means that the network operator may choose to override the MUD file in the manifest. Because the MUD file is not even reported by the device, the network operator is free to do this. The network operator can trust the attestation report because it is signed by the verifier. They trust that the values reported in the CoSWID are accurate because it is signed by the software author who also signs the software. They trust that the device is running the software described in the CoSWID because it matches the attestation report. They trust the MUD file because it is signed by the software author - or because they have supplied that MUD file themselves. MUD files may also be obtained from third-party providers, such as Global Platform Iotopia (<https://globalplatform.org/iotopia/mud-file-service/>).

6. Normative References

- [FDO] FIDO Alliance, ., "FIDO Device Onboarding", n.d., <<https://fidoalliance.org/specs/FDO/FIDO-Device-Onboard-RD-v1.0-20201202.html>>.
- [I-D.ietf-rats-eat] Mandyam, G., Lundblade, L., Ballesteros, M., and J. O'Donoghue, "The Entity Attestation Token (EAT)", draft-ietf-rats-eat-09 (work in progress), March 2021.
- [I-D.ietf-sacm-coswid] Birkholz, H., Fitzgerald-McKay, J., Schmidt, C., and D. Waltermire, "Concise Software Identification Tags", draft-ietf-sacm-coswid-17 (work in progress), February 2021.
- [I-D.ietf-suit-manifest] Moran, B., Tschofenig, H., Birkholz, H., and K. Zandberg, "A Concise Binary Object Representation (CBOR)-based Serialization Format for the Software Updates for Internet of Things (SUIT) Manifest", draft-ietf-suit-manifest-12 (work in progress), February 2021.

- [I-D.ietf-teep-architecture]
Pei, M., Tschofenig, H., Thaler, D., and D. Wheeler,
"Trusted Execution Environment Provisioning (TEEP)
Architecture", draft-ietf-teep-architecture-14 (work in
progress), February 2021.
- [IoTopia] "Global Platform Iotopia", n.d.,
<<https://globalplatform.org/iotopia/mud-file-service/>>.
- [LwM2M] "LwM2M Core Specification", n.d.,
<[http://openmobilealliance.org/release/LightweightM2M/
V1_2-20201110-A/OMA-TS-LightweightM2M_Core-
V1_2-20201110-A.pdf](http://openmobilealliance.org/release/LightweightM2M/V1_2-20201110-A/OMA-TS-LightweightM2M_Core-V1_2-20201110-A.pdf)>.
- [RFC8520] Lear, E., Droms, R., and D. Romascanu, "Manufacturer Usage
Description Specification", RFC 8520,
DOI 10.17487/RFC8520, March 2019,
<<https://www.rfc-editor.org/info/rfc8520>>.
- [SWID] NIST, ., "Software Identification (SWID) Tagging", n.d.,
<[https://csrc.nist.gov/Projects/Software-Identification-
SWID/guidelines](https://csrc.nist.gov/Projects/Software-Identification-SWID/guidelines)>.

Author's Address

Brendan Moran
Arm Limited

EMail: Brendan.Moran@arm.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: 27 January 2022

E. Nordmark
Zededa
26 July 2021

Different aspects of onboarding for IoT/Edge Devices
draft-nordmark-iotops-onboarding-00

Abstract

Previous onboarding discussions have focused on network onboarding. In this note we put that in the context of the larger onboarding picture to also discuss the onboarding to some management or orchestration system.

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 27 January 2022.

Copyright Notice

Copyright (c) 2021 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	2
2. What is onboarding?	2
3. IoT vs. Edge Computing?	3
4. Network onboarding	4
5. Security Considerations	4
6. Example: Project EVE	5
7. IANA Considerations	5
8. Informative References	5
Author's Address	8

1. Introduction

The iotops group has discussed LwM2M [oma], FIDO [fidospec] and [I-D.lear-iotops-deep-thoughts-on-onboarding] where the last one intentionally focuses on network onboarding. This note broadens the discussion to all aspects of onboarding of IoT and edge devices to try to expose what is common and different at different layers.

Some of these topics has previously come up in T2TRG for example in [I-D.irtf-t2trg-secure-bootstrapping] but also with a strong networking focus.

2. What is onboarding?

One aspect of onboarding a device is providing network access to the device. That might involve both L2 and L3 aspects, such as Cellular and WiFi credentials at L2 and LAN as Internet access at L3. Furthermore, the L3 access might differentiate between LAN and Internet access and be subject to access control for instance controlled by MUD [RFC8520].

However, there are also higher levels of onboarding. For instance, Anima supports a notion of Secure Bootstrap over an Unconfigured Network [RFC8994] which not only includes the secure keys (BRSKI [RFC8995]) but also the configuration of the routers and switches (using GRASP [RFC8990]). Such configuration can have rather wide span and one can think of it as consisting of configuring the device plus configuring various applications (which might be routing protocols and management agents in the case of Anima use cases).

If we look at more compute-centric workloads are likely to have a larger set of applications which might be configured and managed separately from the device. We can already see examples of this in cloud datacenters where there is a IaaS layer provisioning and managing the servers, which is largely invisible to the users, and a set of applications (in the form for virtual machines or containers)

which are provisioned and managed using application-specific mechanisms and management systems. For instance, a firewall virtual appliance/VNF might be managed the same way as a physical firewall appliance.

3. IoT vs. Edge Computing?

The IOTOPS charter scopes its use of "IoT devices" to devices that

- * are networked, either to the Internet or within limited administrative domains
- * have a very limited end user interface or no end-user interface at all
- * are deployed in sufficiently large numbers that they cannot easily be managed or maintained manually

The definitions of the various parts of Edge Computing by the Linux Foundation in [lfedge-wp] defines the constrained device edge and the smart device edge, which captures devices with different levels of flexibility, but they both fit into the above IOTOPS scope. Thus for the purposes of this discussion we can use Edge Computing devices and IoT devices interchangeably.

However, the devices at the constrained device edge are more likely to be single or fixed function in that they do not have the capacity or flexibility to perform other functions than envisioned prior to their deployment. Such fixed function devices still require a software/firmware update capability as discussed in [RFC8240], but they do not require handling new application deployment and associated new communication patterns.

The more flexible devices at the smart device edge are likely to be larger than the class 2 devices defined in [RFC7228], however if applications are sufficiently small, constrained devices might very well be edge computing devices. But in general it makes sense to think about devices of the Raspberry Pi class and larger at the smart device edge.

For such devices it is clear that the onboarding of the device (to the network and to some management system or controller) should be separable from the onboarding of some particular application (to its controller or management system). Hence the separation between device onboarding and application (instance) onboarding seems required from an architectural perspective.

4. Network onboarding

Network onboarding starts at L2 access and can take several different forms such as:

- * Physical access to an Ethernet port
- * Protocols like EAP-NOOB [I-D.ietf-emu-eap-noob], DPP [dpp], etc.

In the world of laptop computers and smartphones such access might include traditional EAP but also additional steps such as Endpoint Assessment [RFC5209][RFC7632] before granting full access to the network. Thus the onboarding to the network is not a new thing; what is new is applying it to IoT and Edge Computing devices with to user in front of the device as it is onboarded.

As indicated above, if MUD [RFC8520] is used the network onboarding would logically include the retrieval and application of the usage descriptions.

5. Security Considerations

This informational note discusses onboarding with the assumption that onboarding needs to address various security threats, but does not go into details.

It seems like the roots of trust used for onboarding at the different levels relates closely to the design center for the different onboarding approaches. Loosely we seem to have a few differently approaches (and this list is not exhaustive):

- * Use Hardware manufacturer certificates. This makes it possible to verify with the manufacturer that device is valid, but it does not indicate which management system or controller which a device should trust.
- * Track the transfers of ownership through supply chain as done in FIDO [fidospec]. This enables secure late binding to a management system/controller since the signature chain from manufacturer to end user establishes trust in controller.
- * Imprinting/configuring for/by the owner of the device. This makes assumptions that either the future owner is known at the time of manufacturing or that there is some leap of faith involving a certificate (in e.g., text or bar code form) being registered in the controller by someone claiming to be the device owner.

The trust might also include initial measurement/attestation of firmware/software along the lines of RATS [I-D.ietf-rats-tpm-based-network-device-attest] to create a baseline before the device leaves the factory.

6. Example: Project EVE

Project EVE [eve] is an example of a secure but minimal approach to enable secure onboarding, without having a hard dependency on manufacturers and manufacturer certificates.

* When software is installed (factory or elsewhere):

- Imprint device which controller to trust (a root certificate) and initial URL to contact
- Generate a device certificate using the TPM
- Extract the device certificate and pass to final user (paper, bar code, etc)
- Perform initial measured boot to get baseline measurements along the lines of RATS TPM [I-D.ietf-rats-tpm-based-network-device-attest]

* Then in any order

- User registers device certificate in controller
- Device is installed and powered on and connects to its controller

At that point in time the EVE controller can specify which applications to deploy/boot/halt on device.

Potentially EVE can also leverage [sdo], which is an open source implementation of the FIDO specification [fidospec], for the future cases where there is sufficient support in the supply chain for the FIDO signature chains.

7. IANA Considerations

There are no IANA actions needed for this document.

8. Informative References

- [RFC5209] Sangster, P., Khosravi, H., Mani, M., Narayan, K., and J. Tardo, "Network Endpoint Assessment (NEA): Overview and Requirements", RFC 5209, DOI 10.17487/RFC5209, June 2008, <<https://www.rfc-editor.org/info/rfc5209>>.
- [RFC7228] Bormann, C., Ersue, M., and A. Keranen, "Terminology for Constrained-Node Networks", RFC 7228, DOI 10.17487/RFC7228, May 2014, <<https://www.rfc-editor.org/info/rfc7228>>.
- [RFC7632] Waltermire, D. and D. Harrington, "Endpoint Security Posture Assessment: Enterprise Use Cases", RFC 7632, DOI 10.17487/RFC7632, September 2015, <<https://www.rfc-editor.org/info/rfc7632>>.
- [RFC8240] Tschofenig, H. and S. Farrell, "Report from the Internet of Things Software Update (IoTSU) Workshop 2016", RFC 8240, DOI 10.17487/RFC8240, September 2017, <<https://www.rfc-editor.org/info/rfc8240>>.
- [RFC8520] Lear, E., Droms, R., and D. Romascanu, "Manufacturer Usage Description Specification", RFC 8520, DOI 10.17487/RFC8520, March 2019, <<https://www.rfc-editor.org/info/rfc8520>>.
- [RFC8990] Bormann, C., Carpenter, B., Ed., and B. Liu, Ed., "GeneRic Autonomic Signaling Protocol (GRASP)", RFC 8990, DOI 10.17487/RFC8990, May 2021, <<https://www.rfc-editor.org/info/rfc8990>>.
- [RFC8994] Eckert, T., Ed., Behringer, M., Ed., and S. Bjarnason, "An Autonomic Control Plane (ACP)", RFC 8994, DOI 10.17487/RFC8994, May 2021, <<https://www.rfc-editor.org/info/rfc8994>>.
- [RFC8995] Pritikin, M., Richardson, M., Eckert, T., Behringer, M., and K. Watsen, "Bootstrapping Remote Secure Key Infrastructure (BRSKI)", RFC 8995, DOI 10.17487/RFC8995, May 2021, <<https://www.rfc-editor.org/info/rfc8995>>.
- [I-D.lear-iotops-deep-thoughts-on-onboarding]
Lear, E., "Deep Thoughts on Network Onboarding Challenges", Work in Progress, Internet-Draft, draft-lear-iotops-deep-thoughts-on-onboarding-00, 9 March 2021, <<https://datatracker.ietf.org/doc/html/draft-lear-iotops-deep-thoughts-on-onboarding-00>>.

- [I-D.irtf-t2trg-secure-bootstrapping]
Sethi, M., Sarikaya, B., and D. Garcia-Carrillo, "Secure IoT Bootstrapping: A Survey", Work in Progress, Internet-Draft, draft-irtf-t2trg-secure-bootstrapping-00, 7 April 2021, <<https://datatracker.ietf.org/doc/html/draft-irtf-t2trg-secure-bootstrapping-00>>.
- [I-D.ietf-emu-eap-noob]
Aura, T., Sethi, M., and A. Peltonen, "Nimble out-of-band authentication for EAP (EAP-NOOB)", Work in Progress, Internet-Draft, draft-ietf-emu-eap-noob-05, 12 July 2021, <<https://datatracker.ietf.org/doc/html/draft-ietf-emu-eap-noob-05>>.
- [I-D.ietf-rats-tpm-based-network-device-attest]
Fedorkow, G., Voit, E., and J. Fitzgerald-McKay, "TPM-based Network Device Remote Integrity Verification", Work in Progress, Internet-Draft, draft-ietf-rats-tpm-based-network-device-attest-07, 10 June 2021, <<https://datatracker.ietf.org/doc/html/draft-ietf-rats-tpm-based-network-device-attest-07>>.
- [dpp] Wi-Fi Alliance, "Wi-Fi Device Provisioning Protocol (DPP)", Wi-Fi Alliance , 2018, <https://www.wi-fi.org/download.php?file=/sites/default/files/private/Device_Provisioning_Protocol_Specification_v1.1_1.pdf>.
- [fidospec] FIDO Alliance, "FIDO Device Onboard Specification", December 2020, <<https://fidoalliance.org/specs/FDO/FIDO-Device-Onboard-RD-v1.0-20201202.html>>.
- [oma] Open Mobile Alliance, "Lightweight Machine to Machine Technical Specification: Core", Open Mobile Alliance , June 2019, <http://www.openmobilealliance.org/release/LightweightM2M/V1_1_1-20190617-A/OMA-TS-LightweightM2M_Core-V1_1_1-20190617-A.pdf>.
- [lfedge-wp]
Linux Foundation, "Sharpening the Edge: Overview of the LF Edge Taxonomy and Framework", 2020, <https://www.lfedge.org/wp-content/uploads/2020/07/LFedge_Whitepaper.pdf>.
- [eve] Linux Foundation, "EVE is Edge Virtualization Engine", July 2021, <<https://github.com/lf-edge/eve>>.

[sdo] Linux Foundation, "Secure Device OnBoard", July 2021,
<<https://www.lfedge.org/projects/securedeviceonboard>>.

Author's Address

Erik Nordmark
Zededa
Santa Clara, CA,
United States of America

Email: nordmark@sonic.net

anima Working Group
Internet-Draft
Intended status: Standards Track
Expires: 13 January 2022

M. Richardson
Sandelman Software Works
12 July 2021

Involuntary Onwership Transfer of IoT devices: problem statement
draft-richardson-iotops-iot-iot-01

Abstract

This document details a problem statement relating to ownership of IoT devices.

The problem details is that of changing ownership or possession of a device when against the consent or knowledge of the device and/or manufacturer.

Examples relating to outer door control are used to illustrate the problem statement in an intuitive scope.

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 13 January 2022.

Copyright Notice

Copyright (c) 2021 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	2
2. Door Locks	3
2.1. Human Relationships to Doors and Door locks	3
2.1.1. Single owner	4
2.1.2. Family home	4
2.1.3. Roomates	4
2.1.4. Apartment building	5
2.1.5. Rented or Leased Dwellings	5
2.1.6. Hotels	6
2.2. Rented Automobiles	6
2.3. Additional Third Parties who need access	8
3. Death of a Home Owner	8
4. Multi-person Dwelling: how to kick that that deadbeat roommate out?	9
5. Getting rid of the abusive Spouse	9
6. What is ownership	10
7. Questions and Opportunities	10
8. Privacy Considerations	11
9. Security Considerations	12
10. IANA Considerations	12
11. Acknowledgements	12
12. Changelog	12
13. Informative References	12
Appendix A. Personal Devices	14
Author's Address	15

1. Introduction

Much has been written about how to secure IoT devices against both physical attacks and those that are done through network protocols. (Insert survey articles)

In most cases, the goal of the security mechanisms is to make sure that the device remains under control its lawful or intended owner. One example of such a definition of this control could be to mean that the device accepts commands only from that owner and that the device provides information only to destinations that the owner specifies.

This document explores the problem of what happens when the physical or legal ownership of the device does not correspond to the logical ownership of the device.

There are many ways to explain, scope, and illustrate the general problem. It is much easier to understand with concrete examples, and in this example the front-door lock scenarios are used an easy to understand way to connect to real life intuition. It is believed that most other IoT authorization and ownership problems are probably subsets of the situations outlined here.

2. Door Locks

Most people live in some kind of dwelling with at least one door. When there is more one door, one of them is usually the front-door. This is the primary method of entry and exit, and it usually connects to the street and thus to the rest of the world. It is where both strangers and friends arrive and depart, while other doors (side, garage, balcony, basement and back doors) may lead only areas from which further egress may be impossible, difficult, or deadly.

The door lock is among the simplest of IoT actuator: after potentially many layers of system, there is a single output pin from the lock microcontroller which operates some kind of solenoid. When the solenoid is operated, the door unlocks.

Of course, some doors may be much more complicated with automatic opening or closing motors, sensors to make sure there is clearance before opening, and that the door is clear before closing. Some doors may slide, lift, rotate or perhaps in the future, modulate to alternate dimensions in order to create an opening. None of those details matter to this document.

Also irrelevant to this document are the mechanical details of the door lock itself. While the physical characteristics of the lock are terribly important to actual lock design, it is assumed in this document that the mechanical aspects of the lock is of sufficient quality to resist the expected amount of brute force that is anticipated to be applied to it.

The history of physical door locks is frequent tussle between lock makers who attempt to make locks more resistant to attack, vs thieves who use ever more sophisticated methods to attack the locks. There is an obvious relationship to cryptography and cryptanalysts, and it is hardly surprising that many cryptographers and cryptanalysts are also competent lock pickers. [blazepicking]

2.1. Human Relationships to Doors and Door locks

Homes and apartments come with a complex set of ownership conditions, often via laws established over many centuries. Many places have very ancient laws about when and how a Hotel may evict people.

2.1.1. Single owner

The simplest situation is that of a freestanding dwelling, owned by a single individual.

2.1.2. Family home

To the single individual one adds a spouse, some children of a variety of ages, grandparents, sisters, brothers, neighbours, cat-sitters, etc.

Some members of the household may be trusted to open or close the door from the inside only. For instance a younger child might be allowed to open the door when inside, and only when there is someone else in the house.

The child would not be allowed to leave the house and lock the door, and preventing such an young child from locking themselves out might a useful feature.

Many homes choose to have deadbolts which require a key to lock the door when leaving. Pulling the door shut is insufficient to lock the door.

Other owners prefer that the door lock itself when pulled closed, and so might use a spring-bolt lock.

Still others have double deadbolts which require a key in the inside in order to lock or unlock the door. People prefer these if they are concerned that a thief will enter their home through a window, and then will go out the front door with their stuff. The double deadbolt requires a key to unlock from the inside. The downside of the double deadbolt is that in the event of a emergency, it is not possible to use the door without the key. As a result, many homes with a double deadbolt will have a key hanging nearby, but not within reach of a window.

2.1.3. Roomates

One scenario where there are multiple unrelated individuals in a dwelling is when it is shared by roomates. Each roommate will have co-signed the lease and will have an equal right to be in the apartment. It would be inappropriate for any roommate to have the power to lock out the other roomates.

This is contrasted with a owner (or renter) who sublets one or two rooms to other people. In that case, this primary owner should have more power over who can enter and exit, subject to some legal restrictions. The degree to which subletter have legal rights varies by jurisdiction.

Can any of these individuals give a "key" to girlfriend/boyfriend? This is definitely a complexity of the situation which is usually not seen in the family home.

2.1.4. Apartment building

An apartment building consists of many dwellings with some common space. (This is distinguished from a multi-tenant building where each tenant has their own front-door.)

Residents of an apartment buildings must pass through a common front door. Historically access to such buildings was via a kind of guard, the door-man. This has now been replaced with some kind of master-key on the front-door, which a telephone mediated system that allows visitors to "buzz" up to the appropriate apartment. The resident of that apartment then activates a circuit to unlock the front door.

Historically, these telephone systems were hardwired private handsets present in each apartment. This meant that anyone who was in the apartment could let anyone else in.

More modern system are tied into the public telephone system, and a DTMF tone is used to unlock the front door. With such a system, if the phone number attached to the apartment is a mobile phone, then a resident can buzz themselves while outside the apartment, and then buzz themselves in.

The modern apartment system does not usually provide for multiple numbers to be attached to the system, and a guest in such an apartment would be unable to, for instance, let medical people in, if the primary resident took ill.

2.1.5. Rented or Leased Dwellings

Many dwellings are owned by one person, but occupied by another person based upon a rental agreement.

Historically such agreements were based upon leases of many months to years, but intermediation of the relationship by a number of dotcom companies have reduced the lease time to days, and the same rental systems are expected to accomodate what is more like a Hotel relationship. That situation is handled in the Section 2.1.6 section.

In many cases the owner (or property manager) of the home has a legal right to enter, under certain circumstances. For instance to effect repairs, to show the dwelling to a new potential tenant, and in emergencies, to do things like shut off water or gas to avoid damage.

Notice is often required for most activities, most laws allow a landlord to enter without notice during emergencies to do things like shut off water when there is a leak. A landlord can also be compelled to open the door for a police warrant, and in cases where the police suspect harm, they often will enter without a warrant.

This situation is even more complex in apartment buildings, even where the apartments are owned (and occupied by the owners). There is still a building manager, and there are still water leaks.

There is additionally, many common areas to which many people should get access. Some areas like common rooms are multi-access, but during a reserved time, are exclusive to the person who made the reservation.

Additionally, there are secondary areas that are private to each residents, such lockers for bicycles and parking spaces.

2.1.6. Hotels

Placeholder.

2.2. Rented Automobiles

Automobiles have doors, locks, and ignition locks. There are sometimes different keys for the different locks. The valet key for instance, allows the driver door to be opened and the car to be started, but does not provide access to the glove compartment or the trunk.

Automobiles are rented in a variety of ways: from hourly rentals by car-sharing companies (e.g., [communauto], [zipcar], [tribecar]..), to traditional daily rentals by well-known companies, to yearly car leasing.

During the valid period of rental, the motorist probably needs to have complete control of the vehicle. If any other party had any control of the vehicle, it might significantly change the legal liability for activity done with the vehicle.

This is usually done by giving them a key which they must insert into the ignition.

Some car sharing companies have schemes involving lockboxes (with master physical keys!) to share the car-specific key. (This is rather akin to Kerberos tickets: one key is used to unlock another key)

Increasingly automobiles are going "keyless", and it is sometimes sufficient for the "fob" to be just near the vehicle, but the fob is essentially still a key.

Many manufacturers are now using the individual's smartphone to unlock the car via Bluetooth or NFC, and once inside the vehicle, the phone serves as the "fob", authorizing the vehicle to run.

Integration with the smartphone has a transaction cost to it: the phone/car connection must be onboarded in some way, and is therefore only suitable for car owners, or longer-term leases.

Shorter term leases may transition to use of a smartphone, but today, they are mostly based upon passive RFID FOBs or physical keys. Today, when used via smartphone, there is a satellite or LTE based care security system that the drive interacts with via the Internet. There are reports of people being stranded in the woods for days, because they were too far away from the LTE tower, and the vehicle would not unlock or start without authorization.

At the end of the rental period, the access for the motorist must be revoked. This is akin to getting rid of roommate (Section 4). But there are some caveats: there has to be some kind of grace period or interlock with the renting agency, as the vehicle might not yet have been returned properly. They could just be late. The vehicle could stall meters from the proper location and need to be restarted. Once at the proper location, the motorist might still need to access the trunk or other compartments to retrieve their belongings.

But, once properly returned, the vehicle should no longer be accessible to the original renter.

The next renter may be standing waiting, particularly if the vehicle is late. The transition from one renter to another needs to have a standardized ceremony.

For long-term leases the process may be more complex at the end. While some significant grace period (compared to rental period) is appropriate for short-term, for longer term leases, the owner likely needs to be able to disable the vehicle some few number of days after the end of the lease. But, never before.

2.3. Additional Third Parties who need access

In addition to this obvious arms race, there are specific third parties that bring their own interests to the locks in the front-door lock scenario, e.g. law enforcement or fire departments.

In some places there are locks which accept keys carried by fire, police or postal personnel. For instance, the service key in a building allows the fire department to override the elevator controls. The electrical panels and gas systems in the buildings may also be accessible by the fire department in order to cut off electricity or gas during a fire.

The mailboxes of an apartment (and the outer door to get to the mailbox) can be opened by the postal carrier in order to deliver the residents mail. The French PTT T-10 key is an example of such a key, and there is a law and regulation around it as well.

This is an example of a master key necessary in most multi-tenant buildings.

It is hardly surprising that there was significant concern when the fire/police "master key" for the city of New York was being openly sold on ebay. (see [huffpostkey] and [fdnymaster])

A digital door (and elevator control) key that could be safely deployed as a replacement for this physical key would be a significant improvement over the physical keys. It would be easier to add new users and revoke old users, and an audit log of who used what key in which building could be easily generated.

3. Death of a Home Owner

Start from a single freestanding dwelling, owned by a single individual, and ask what happens when the individual dies. How do the inheritors (or the executors of the estate) take possession of the property? Prior to electronic door locks, a physical key can be used, and if one is not available, then a locksmith can be engaged. This may require a legal statement from an appropriate authority, at which point the locksmith may make use of a drill, or maybe even some other implements such as saws or battering rams.

The same techniques can probably be used against electronic door locks that do not use keys, but can this technique be used against, for instance, smart toasters, furnaces or automobiles?

Repairing a hole in a front door is a nuisance. Replacing a furnace or other large appliances due to a death is unacceptable.

In particular, automobile locks are usually designed to resist significant amounts of force as they are often the target for thieves. The vehicles are left unattended in public parking lots among many other automobiles for many hours at a time, and it is even a common occurrence that a person legitimately walks up to the wrong automobile (having forgotten exactly where they parked) and attempts to unlock it.

Any tool or protocol that the locksmith can employ against the automobile could also be employed by a malicious attacker. Any mechanism that the automobile maker includes in the system to allow a locksmith (or legal court) to open the vehicle would be the target of attackers. This is fundamentally why security protocols do not include back doors ([RFC1984]).

4. Multi-person Dwelling: how to kick that that deadbeat roommate out?

The situation above was for a single dwelling. Many dwellings are occupied by multiple people, often jointly.

Should any of the occupants be allowed to change the locks, that is, change the entry authorization for other occupants? Under normal circumstances, the answer should probably be no. Under the situation of a legal injunction, the answer may be yes. How can the door lock system know? How can the party which is asking for the injunction know that the door lock has no other secret authorizations?

If the legal system must be a party to this activity, how does the home owner, not involved in such a process know that the legal system's computers haven't itself been compromised? This is one of the major arguments against official escrow: the escrow system is now a very high value target.

5. Getting rid of the abusive Spouse

The situation where a couple separate under duress requires that access to the original home be restricted. That is, the door locks must be rekeyed. Digitally, this means removing the access to the abusive spouse.

Is this different than the case of roommates? Not really: multiple people had access to the door lock before, and one must be removed. For the case of roommates, each had a legal right to access, and no roommate should be allowed to revoke access for the other roommate.

Now, in the case of separation, the remaining "roommate" must now be permitted to revoke access for the other "roommate"

6. What is ownership

One technical definition of ownership might be that the device has an identity certificate from the owner. This is a good definition, and it is currently what is used in [RFC8995], [MATTER], and many other similar systems.

In the security space, the vernacular term, "p0wned" is often used to refer to a device that is no longer under the control of the legitimate owner. That is, an attacker has taken control of the device, usually through some security vulnerability, and now the attacker controls what code the device will run.

So a deeper notion of what it means to own a device is that it could involve control of what software a device runs. Whomever controls the software in a device controls what the device does, and whose commands it obeys. This can generally be expressed in the form of an authorization from a Trust Provisioning Authority (Section 7 of [RFC9019]).

Control and access decisions are not usually changed by changes to the firmware of the device. (Notwithstanding the dispute between the FBI and Apple: [applefbi]) For good or bad, all devices of a particular type run the same firmware that the manufacturer has provided. The decision as to who is in control of the device is determined by the firmware based upon the identities of the parties.

All of the challenges in the previous section boil down to finding a way to express the question as to whether an identity is allowed control.

7. Questions and Opportunities

While the example of the front door lock was used as an exemplar, essentially the same question applies to pretty much all forms of actuator. Access to some sensors may be significantly simpler, but other sensors will be as complex as any actuator.

A primary question is whether the front door problem is a superset of all other problems. If so, then a solution to the front door ownership can provide for all other actuators.

Or, if there some other physical world interaction which is more complex, then the front door may be a subnet of it. Alternatively there may be some other master pattern which does not overlap with the front door and it would provide a different model. Some actuators might be a subset of these two models.

The various modes of front door interaction need to be named. Based upon the above description, these would include: roommates, spouses, ex-spouses, renter/owners, tenant/superintendent, fire-department, police officer, young-children/parent, adult-children/seniors...

The automobile, personal or medical device interactions are mostly variations on the front door. Instead of superintendent, substitute mechanic, leasor or ER doctor. Instead of child, substitute neighbour-who-borrows your tools.

The IETF has created a number of authorization systems. This starts with SPKI [RFC2393], OAuth2 [RFC6749], Authorization in Constrained Environment [RFC7744], SAML ([oasissaml] and [RFC7522]). There are many others: most are based on the providing virtual access to a virtual resource (computer, web resource,etc.) rather than authorizing physical access to a physical resource.

Can the required policies be representing in the existing frameworks? If so, are the frameworks we have sufficiently small as to live within a front door lock? Perhaps a better question is: what is the price point that society is willing to pay for a front-door system which satisfies the various needs of the multitude of stakeholders involved?

8. Privacy Considerations

There is a significant tussle between having policies which are clearly asserted (and auditable) and having privacy for the individuals or groups named.

For instance, it may be entirely appropriate for a front door to make it clear who is allowed access in the event of emergency, such that those people can easily be found. On the other hand, it may be inappropriate for the front door to list one's current romantic interests as having access. (Such access might even be "aspirational")

A significant mix of abstract identities ("The Superintendant of the Building"), along with pseudonymous identities will be required.

9. Security Considerations

This entire document is about a proposed set of authorization systems.

10. IANA Considerations

This documents makes no IANA Requests.

11. Acknowledgements

Hello.

12. Changelog

13. Informative References

[applefbi] "Apple, Americans, and Security vs. FBI", n.d.,
<<https://www.eff.org/deeplinks/2016/02/apple-americans-and-security-vs-fbi>>.

[blazepicking]
Blaze, M., "Notes on Picking Pin Tumbler Locks", 7
November 2003,
<<https://www.matthblaze.org/papers/notes/picking/>>.

[communauto]
"Communauto Car Sharing", n.d.,
<<https://www.communauto.ca/>>.

[fdnymaster]
Schneier, B., "Schneier on Security: Master Key", 10
January 2012,
<https://www.schneier.com/blog/archives/2012/10/master_keys.html>.

[huffpostkey]
Huffington Post, "Daniel Ferraris, Retired Locksmith,
Sells NYC Master Keys On eBay", 10 January 2012,
<https://www.huffpost.com/entry/daniel-ferraris-new-york-master-keys_n_1928826>.

[MATTER] Alliance, C.S., "Connected Home over IP Specification", 1
July 2021, <<https://buildwithmatter.com/>>.

- [oasis-saml] "OASIS Security Services (SAML) TC", n.d.,
<https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=security>.
- [RFC1984] IAB and IESG, "IAB and IESG Statement on Cryptographic Technology and the Internet", BCP 200, RFC 1984, DOI 10.17487/RFC1984, August 1996, <<https://www.rfc-editor.org/info/rfc1984>>.
- [RFC2393] Shacham, A., Monsour, R., Pereira, R., and M. Thomas, "IP Payload Compression Protocol (IPComp)", RFC 2393, DOI 10.17487/RFC2393, December 1998, <<https://www.rfc-editor.org/info/rfc2393>>.
- [RFC6749] Hardt, D., Ed., "The OAuth 2.0 Authorization Framework", RFC 6749, DOI 10.17487/RFC6749, October 2012, <<https://www.rfc-editor.org/info/rfc6749>>.
- [RFC7522] Campbell, B., Mortimore, C., and M. Jones, "Security Assertion Markup Language (SAML) 2.0 Profile for OAuth 2.0 Client Authentication and Authorization Grants", RFC 7522, DOI 10.17487/RFC7522, May 2015, <<https://www.rfc-editor.org/info/rfc7522>>.
- [RFC7744] Seitz, L., Ed., Gerdes, S., Ed., Selander, G., Mani, M., and S. Kumar, "Use Cases for Authentication and Authorization in Constrained Environments", RFC 7744, DOI 10.17487/RFC7744, January 2016, <<https://www.rfc-editor.org/info/rfc7744>>.
- [RFC8995] Pritikin, M., Richardson, M., Eckert, T., Behringer, M., and K. Watsen, "Bootstrapping Remote Secure Key Infrastructure (BRSKI)", RFC 8995, DOI 10.17487/RFC8995, May 2021, <<https://www.rfc-editor.org/info/rfc8995>>.
- [RFC9019] Moran, B., Tschofenig, H., Brown, D., and M. Meriac, "A Firmware Update Architecture for Internet of Things", RFC 9019, DOI 10.17487/RFC9019, April 2021, <<https://www.rfc-editor.org/info/rfc9019>>.
- [tribecar] "Tribe Car", n.d., <<https://www.tribecar.com/>>.
- [zipcar] "ZIP Car", n.d., <<https://zipcar.com/>>.

Appendix A. Personal Devices

There is an increasing number of devices that a person might have on their person or around them. The list is endless, and goes from step trackers, to watches, to recreational (exercise) heart monitors, shoes, shirts with displays (for fun or for the disco), to intimate devices that might be worn at unusual times.

Some devices may belong only temporarily to a person. For instance, a tread-mill or weight-lifting machine, or even a kitchen appliance. After the user is finished with the device it may need to reset to be ready for the next user.

A kitchen appliance (a blender or microwave) might have only a small number of legitimate users (the members of the household), but when one person is using it, it might remain exclusive.

The same appliance, however, might also be purchased for use in a workplace kitchen, and so the number of legitimate users might range in the hundreds. The users will want the appliance to remember their personalized settings.

The names of the previous users should not be easily divulged, but at the same time, the name of the person who used it should be available to a privileged user (owner), for the case the finding out who broke the device. In this case, it might seem obvious that the device has a privileged owner, and may also have just users. But this interaction may be quite complex, and is subject to a wide variety of locally significant social compacts.

In addition, devices get lent. This could be akin to thinking about there being users vs owners, with the owner always being the one responsible for the device. However, passing on a coffee maker to one's child who is moving to another city is not always a loan, and not always a gift. Which one it is may not be obvious to the people involved until later on. The parent may forget about it, thinking they have given it away, while the (adult) child might pass it on to a friend. Only when the friend tries to "own" the device, do they find out that the parent is still the owner. Now what? Does the device have to be returned to the parent to physically give away ownership?

If the answer to the above question is no, then does this in essence enable theft? Is this a kind of theft that we need to care about? Does it matter if this is a \$50 coffee maker, vs a \$600 espresso machine? Or can we even set a meaningful threshold? Theft of a \$600 espresso machine might not be a problem for some people, while the loss of a \$50 coffee machine might be a rather big problem.

Author's Address

Michael Richardson
Sandelman Software Works
Email: mcr+ietf@sandelman.ca