Driving network video through global standardization

# ONVIF Security Recommendations

## White Paper

Version 1.0

March 2010

**Author:** Christian Gehrmann,

Axis Communications AB and

Swedish Institute of Computer Science

# 1 Background

Security is a very important aspect to consider for network based services. This is true for ONVIF services as well as other IP network applications. Security covers everything from protecting communication between devices and the computing platforms to secure device management functions and end user interfaces. The ONVIF specifications [1] include basic mechanism for securing communication on the transport and message levels. These mechanisms have been introduced to allow interoperable communication and configuration between ONVIF clients (NVCs) and ONVIF transmitters (NVTs). The mechanisms in the specifications only give some "general hooks" for secure configuration and communication. A full system security solution requires much more. In this white paper we give some general security recommendations to assist network video product manufactures, integrators and system administrators in producing and configuring devices such that a sufficient network video system security solution can be achieved.

# 2 General security recommendations

The ONVIF security mechanisms are *only enablers* for interoperable transport and message level security and far from a *complete* network video security solution. The protection mechanisms provided by a particular ONVIF compliant product is expected to vary a lot and the customer should choose a product that best fits the security needs for the particular network/application.

The ONVIF *standard* security mechanisms describes transport and message level protection means. These should be used with care and additional protection measures should be considered in relation to these. In particular we will discuss the following issues:

- NVT software and hardware platform security
- Credential handling
- Security policy and sensitive ONVIF commands
- Media streaming protection

## *2.1 NVT software and hardware platform security*

The TLS protocol [2] provides peer authentication and protected transport of all information exchange (above TCP level) between the NVC and NVT. Similar, the ONVIF Web Services security protects the ONVIF command exchange between the NVC and NVT. The security of the TLS and ONVIF command layers are dependent on that reliable TLS software *and* that reliable OS are executing on the device. If not, there is risk that hostile software on the device intercept decrypted data, session keys, user credentials etc. and forward such information to a third party or that the illegal software destroys essential security functions in the system.

To guarantee the security of the NVT or NVC software platform is very hard as long as the platforms also should be *open* for software/hardware upgrades which are essential requirements on most systems. Hence, often there needs to be a tradeoff between the desired security level that can be offered on platform level and the flexibility regarding software upgrade that is needed. Strict control of all software upgrades and regular integrity checks of the software platforms are mechanisms that improve the security level. Special purpose security hardware that assists in protecting the software platform is a useful tool to improve the platform security level. Example of such hardware is a Trusted Platform Module (TPM) according to the Trusted Computing Group (TCG) standards [3]. A TPM can be used both on a high-end server platform and on more restricted/embedded platforms.

### *2.2 Credential handling*

Both message and transport level security heavily depends on secure handling of user and/or device credentials. In Section 3 and Section 4 below, we give some recommendation on how to protect the credentials.

### *2.3 Security policies and sensitive ONVIF commands*

Protecting *all* ONVIF services with a *single* protection profile is often not a particular good idea as that will not allow *less restrictive* security requirements on services that should be available to *many* different users or devices in the network. Hence, there is a need to define what type of message and transport protection that should apply to different type of services. Such definition is often called a *security policy*. The first version of the ONVIF specification does not contain any standardized security policy format. Instead, the format is assumed to be unique per NVT manufacturer. In the next version of the specification we expect an ONVIF *standardized* security policy format to be in place.

Independent of if the format is standardized or not, it is important to provide an NVT security policy that in a good way reflects the security needs in the network. In many networks a security policy build on the ONVIF standard mechanisms TLS with server side authentication and the username token profile will give a *rudimentary* security level with mutual authentication that *might* be enough considering the threat situation. Such a security policy could either require TLS with server authentication to be used on *all* commands or just a subset, i.e., the most security critical.

This is then preferably combined with authentication requirement on username token level. To be flexible enough, the different ONVIF user categories [1] in such configuration, i.e., administrator, operator, user and anonymous, are mapped into the different ONVIF services and commands through the defined policy. One of the reasons why such configuration only give basic protection is that username tokens *do not* provide strong replay protection and the security strength will depend directly on the length of the user password etc. Also one should avoid using *the same* password for many different NVTs as that makes the whole network much more vulnerable to attacks (see the recommendations in Section 3.1).

In a network with higher security requirements, either TLS with server authentication in combination with *a strong message level protection mechanism*, such as X.509 or Kerberos token can be used. An alternative is using TLS with *both* server and client side authentication which can be complemented with message level (WS-Security) authentication requirements *or* providing confidentiality and integrity protection on WS-Security level (such option will not give any protection of streaming media though).

Some ONVIF commands are obviously more sensitive to attacks than other commands. In particular, the ONVIF device management service is very sensitive and need high protection. A flexible ONVIF security policy gives strong protection on the most sensitive commands while less sensitive commands can be left open to anonymous or a large authorized set of users.

### *2.4 Media streaming protection*

Depending on the vulnerability of the network, protection of the media content itself can be of high importance or not. Obviously, in closed and controlled private networks, there is not so much need for protection of media content and streaming control. In completely or partly open or large networks, media protection is essential. The ONVIF standard [1] defines media transport protection based on TLS (HTTPS) that should be used in all vulnerable environments.

Also media control based on RTSP needs to be protected to prevent attacks on existing media streams. The following is *strongly recommended:*

- Authentication shall always be done on RTSP level (and not on HTTP level when RTSP is tunneled over HTTP).

The RTSP authentication should be carried out using username/password *digest* authentication [6] according to [5]. For interoperability reasons, when a WS-security username token is used to setup

the media configuration on the device, the same username and password as used for the username token, should be used also to authenticate the RTSP requests.

# 3 Token handling

As we discussed in Section 2.3, several ONVIF commands are sensitive to sever attacks on the network video system. Hence, it is *very* important to protect the ONVIF control exchange. At the same time, the security requirements should not make the system complex and hard to administrate. To find the right balance between these two contradictions, is a well known and hard task in all security systems. ONVIF requires all NVTs and NVC to support the WS-Security username token [4]. This allows the offering of *interoperable* message level security between ONVIF devices. However, the username token only gives a *rudimentary* protection level. When *combining* username token based command authentication with transport level security, sufficient security can be achieved in systems without very strict security requirements. As the username token as such does not provide message integrity or confidentiality *and* it does not provide any strong replay protection mechanism, it must be used with care. In Section 3.1, we give some recommendations for implementers and security managers with the purpose of giving advice that can assist in avoiding some security pitfalls.

Higher security level, including integrity and confidentiality protection, can be achieved by choosing more secure tokens. In Section 3.2 we give some security recommendations regarding the X.509 and Kerberos tokens.

## *3.1 Username token*

The username token security is based on so called usernames and corresponding passwords. In addition the ONVIF standard has introduced four *user levels*, administrator, operator, media user and anonymous. Each user should be assigned an ONVIF user level in order to make *authorization* decisions for different users. Preferably, each NVT has protected access to a *database* where all users, corresponding password or password equivalent and authorization level are stored. Depending on the system set-up, this database can be an NVT *local* database, or a *centrally* database on a protected server in the network. The latter has the advantage that higher protection of users and their passwords can be provided. Similar, it is easy to revoke users. The main drawback is that it requires special *customization* procedure of the NVT to *securely* associate it with the central

user database and the current ONVIF specification does not include any standardized procedures for making such configuration or for securing the connection to the database. Furthermore, it requires a central sever in the system for user handling and to be secure, each authentication request should be forwarded to the central server which can cause a *considerable* communication overhead and delays. Hence, unfortunately, for most early ONVIF system using the username token, one can expect an NVT *local* user credential database. Such configuration implies several security risks. Below, we give some general recommendations to on how improve the security and at the same time provide better interoperability in systems with NVTs from different manufacturers.

### 3.1.1 NVT username token bootstrapping

At the time of shipping, the NVT must be preconfigured with some security policy and *preliminary* credentials. We recommend the following:

- All NVT services *except* the device service are by default *disabled.*
- The *only* ONVIF command that will be accepted from scratch by the NVT is the "CreateUser" command and the NVT first time user is expected to run this command and to create a new administrator user. Once this is done, the NVT will *enable* the rest of the ONVIF services.
- The NVT will then apply the *default* NVT security policy on all offered services. The default security policy is for the first version of the ONVIF specification *manufacture* specific.
- It is strongly recommended to *only* accept user creation or update requests from users with administrator role *and* that the *default* security policy include such configuration

### 3.1.2 Username token password derivation

In large to mid size network video systems, it would be very inefficient to require *different* username and passwords for each NVT in the system (from the NVC perspective). On the other hand, if *many* NVTs in the system have *common* user credentials, i.e., one user and corresponding password is accepted and *stored* in two ore several NVTs, implies that compromise of a *single* NVT in the system will destroy the security of the rests of the NVTs in the group. Similar, it opens up for replay attacks as the same command will be accepted on all NVTs in the system clocks within the acceptance window. However, the WS-Security username token specification [4] does not require that the "password" used for user authentication actually is the "user password" (used and remembered by the end user), but it can be a so-called "password equivalent". By using password equivalents instead of pure user passwords, the security threat above can be addressed. This implies

that *instead* of storing the “user password” when creating or modifying a new NVT user, *an NVT unique* password is derived using the “user password” and a NVT unique identifier. A suitable such NVT identifier is the end point reference value. This implies that the NVC needs to *derive* the NVT unique password each time a new NVT user is created and when an end user presents his/her user credentials to the NVC.

If each system uses its own way to derive ONVIF username token password equivalents, there will not be any interoperability between NVCs from different providers. Hence, we recommend NVC to implement the password calculation algorithm defined below.

Denote by UA an arbitrary user. Denote by P_UA the password value used by user UA to access the NVTs in the system. Furthermore, denote, by NEP, the end point reference value for a particular NVT in the system in its *binary form*, i.e. the 128 bit unique endpoint identifier value. Finally, denote by PE_UA the password equivalent used by the NVC to access a particular NVT in the system. The NVC should calculate the PE_UA as follows:

PE_UA = base64(HMAC_SHA-1(P_UA,NEP+”ONVIF password”)),

where “+” denotes concatenation and where the “ONVIF password” is an ASCII string. It should be included in the exact form it is given without a length byte or trailing null character, i.e., the following hexadecimal value: 4F 4E 56 49 46 20 70 61 73 73 77 6F 72 64.
HMAC_SHA-1 is the algorithm specified in [7] using SHA-1 [8] as the underlying algorithm. The key value to use for the HMAC function is the user password, P_UA, directly mapped to its binary equivalent. Similar, the value PE_UA should be mapped to its ASCII equivalent before transmitting it to the device. Base64 is described in [9], note that the result of the base64 operation is the actual password equivalent and shall be used as it is.

#### 3.1.2.1 Example

Assume the following password is used by the NVC(ASCII): “VRxuNzpqrX”, i.e.,

P_UA = 56 52 78 75 4E 7A 70 71 72 58

Next, assume the NVT has the following end point reference value:

Urn:uuid:f81d4fae-7dec-11d0-a765-00a0c91e6bf6.

Then the password equivalent to be used will be then calculated as:

PE_UA = base64(HMAC_SHA-1(P_UA,NEP+"ONVIF password")) =
base64(HMAC_SHA-1(565278754E7A70717258, F81D4fAE7DEC11D0A76500A0C91E6BF6+
4F4E5649462070617373776F7264)) =
base64(7E07971EA430EEA591C1931BFA6FF6B67E340290) =
fgeXHqQw7qWRwZMb+m/2tn40ApA=

The resulting password equivalence "fgeXHqQw7qWRwZMb+m/2tn40ApA=" is the password that shall be used by a client both for configuring the user credential on the particular device and then also for accessing the device.

### *3.1.3 Username token timing issues*

The WS-Security username token specification [4] states that used nonces should be cached for a period at least as long as the timestamp freshness limitation period and that username token with nonce that have already been used (and are thus in the cache) be rejected. Hence, it is important that the NVT keep track of used nonce values *at least as long* as the timestamp freshness period. Especially, if nonces are only cached in volatile memory, the NVT should back up the nonce values at reboot or system crash (if possible).

In order to work well, it is important the NVCs are allowed to get NVT time information *without* the need to authentication. Consequently, we recommend allowing anonymous access to the NVT GetSystemDateAndTime system operation.

The timestamp freshness period should be adjusted to the available nonce cache size. The NVT should not accept a longer freshness period than can be allowed according to the nonce cache size.

When the creation time is outside the time window, the NVT should return an wsse:FailedAuthentication error code as defined in [10]. It is important the NVC time is *synchronized* with the NVT time in order to provide correct digest authentication.

### *3.1.4 Username token database protection and maintenance*

NVT local username token database should be adequate protected from network, software and direct physical attacks. Hence, it *is not recommended* to store the database in clear text on any non-volatile memory in the system. Also, if the database (in parts or completely) is temporarily stored in

clear on volatile memory, protections means that prevents direct read out, or read out from hostile software, should also be in place. Preferably, the database is stored in protected hardware. *In addition,* it should only be accessible by software running in a protected execution environment[1].

### *3.2 Other tokens*

*If* higher security is needed or if TLS for some reason not is feasible to use, one have the option to use a *stronger* message level security mechanism. The ONVIF standard recommends using strong WS-Security tokens whenever applicable. Tokens such as the X.509 [11] or Kerberos [12] token enables strong message level security options including integrity and confidentiality of ONVIF commands.

The X.509 token can be used to transfer information such that the SOAP message can be signed and/or encrypted. If this should work properly, there must be means for the receiver to *securely* verify the trustworthiness of the included or referenced X.509 certificate. This in turn requires that the NVT and NVC respectively *trust* the certificate issuer. Hence, secure configuration of trusted certificate issuer needs to be in place in order to provide an adequate security level. Integrity X.509 protected messages requires digitally signing/verification of asymmetric signatures which can be a rather computational demanding task. Similar, *asymmetric* decryption/encryption of *symmetric* encryption keys can also be rather computational demanding. Consequently, these protection methods are not always well suited for protecting of command exchange with real-time requirements such at PTZ control. For these commands, symmetric based authentication such as Kerberos token based protection, is a better choice (see below).

Kerberos tokens can be used in network configuration were a *Kerberos Key Distribution Centre* (KDC) is available. Strong confidentiality and integrity protection can be provided on SOAP message level in a Kerberos based system. The major drawback with Kerberos based protection is that each NVT and NVC is the system needs to be *securely* associated with the KDC, which can be a rather demanding security management task. On the other hand, Kerberos protection is rather well suited for protection also of commands with strict real-time requirements.

[1] The need for protected execution environment will depend on how “open” the platform is. A closed, tightly controlled software platform would fulfill the secure execution environment requirement.

# 4 Transport security handling

In order to provide secure transport protection, the NVT and NVC must be correctly and securely configured. This particular relates to secure configuration of client and server certificates, revocation information and trusted Certificate Authorities (CAs). The first version of the ONVIF standard includes a set of management commands for NVT certificate administration. The enables interoperable:

- On-board generation of NVT key pair (creation of self-signed certificate)
- Deletion of certificates
- Issuing of new server certificates through PKCS#10 [13] requests
- Loading of new server certificates
- Enable/disable of NVT client authentication

This is only *a basic* set of TLS certificate management commands. In future versions of the specification we expect a richer set of commands to be standardized that would allow more advanced configurations. In particular, currently the following need to be configured using manufacturer specific procedures:

- NVC trusted CAs, for verification of NVT server certificates
- NVT trusted CAs, for verification of NVC client certificates
- Client certificate issuing
- Certificate revocation mechanisms

In ONVIF 1.01, depending on the network the NVT will be deployed in, one can think of several different ways for configuration of trusted CAs. Either, this is done at a *customization* procedure prior to the shipping of the NVT product or the NVT manufacturer provides a proprietary interface (local or Web Service based) that can be used to set trusted CA roots in the device. Most VMS or PC platforms already have well defined interfaces for NVC CA configuration. Similar, there are lots of tools available for certificate issuing on these kinds of platforms.

For certificate revocation (for both client and server certificates), there already exist widely deployed protocols such as OCSP [14] that we recommend to use also for ONVIF entities.

The NVT private keys and trusted CA configurations are *very sensitive* security configurations that need to be well protected. Preferable private keys are only exposed to secure execution environments protected by hardware. The keys and configuration should be stored in protected hardware or in integrity *and* confidentiality protected non-volatile memory,

# 5 References


[1] ONVIF Core Specification, ver. 1.01, 2009. URL:http://www.onvif.org

[2] T. Dierks and E. E. Rescorla,"The Transport Layer Security (TLS) Protocol Version 1.2" , August 2008. URL:http://www.ietf.org/rfc/rfc5246.txt

[3] Trusted Computing Group, URL:http://www.trustedcomputinggroup.org/

[4] "Web Services Security UsernameToken Profile 1.0", OASIS Standard, March 2004. URL:http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0.pdf

[5] H. Schulzrinne, A. Rao and R. Lanphier , "Real Time Streaming Protocol (RTSP)",RFC 2326, April 1998. URL:http://www.ietf.org/rfc/rfc2326.txt

[6] Franks, J., Hallam-Baker, P., and J. Hostetler, "An extension to HTTP: digest access authentication", RFC 2069, January 1997. URL:http://www.ietf.org/rfc/rfc2069

[7] H. Krawczyk, M. Bellare and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, February 2007. URL:http://www.ietf.org/rfc/rfc2104

[8] Federal Information Processing Standards Publication 180-2, "SECURE HASH STANDARD", August 2002. URL:http://csrc.nist.gov/publications/fips/fips180-2/fips180-2.pdf

[9] S. Josefsson (Ed.), "The Base16, Base32, and Base64 Data Encodings", RFC 3548, July 2003. URL:http://www.ietf.org/rfc/rfc3548

[10] "Web Services Security: SOAP Message Security 1.0", OASIS Standard, March 2004. URL:http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf

[11] "Web Services Security X.509 Certificate Token Profile 1.1", OASIS Standard, February 2006. URL:http://www.oasis-open.

[12] "Web Services Security Kerberos Token Profile 1.1", OASIS Standard, ,1 February 2006. URL:http://www.oasis-open.org/committees/download.php/16788/wss-v1.1-spec-os-KerberosTokenProfile.pdf

[13] PKCS #10 v1.7: Certification Request Syntax Standard, RSA Laboratories, May 2000.

URL: ftp://ftp.rsasecurity.com/pub/pkcs/pkcs-10/pkcs-10v1_7.pdf

[14] M. Myers et al., "X.509 Internet Public Key Infrastructure Online Certificate Status Protocol – OCSP", RFC 2560, June 1999. URL:http://www.ietf.org/rfc/rfc2560.txt