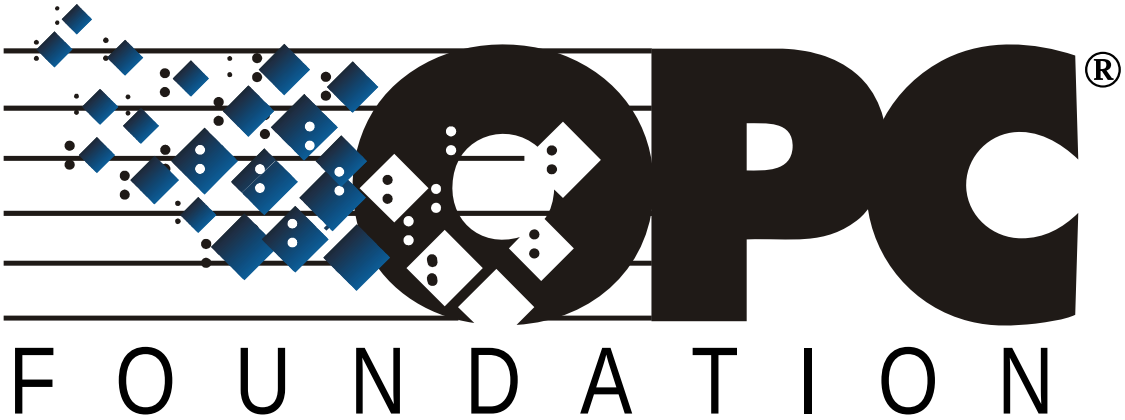

OPC Unified Architecture

Specification

Part 2: Security Model

Version 1.00

July 28, 2006

CONTENTS

1 Scope

This specification describes the OPC Unified Architecture security model. It describes the security threats of the physical, hardware, and software environments in which UA is expected to run. It describes how UA relies upon other standards for security. It gives an overview of the security features that are specified in other parts of the OPC UA specification. It references services, mappings, and profiles that are specified normatively in other parts of this multi-part specification. This part of the specification is informative rather than normative. Any seeming ambiguity between this part and one of the normative parts does not remove or reduce the requirement specified in the normative part.

This Part 2 is directed to readers who will develop OPC UA client or server applications or implement the OPC UA services layer.

2 Reference documents

2.1 References

[UA Part 1] OPC UA Specification: Part 1 – Concepts
http://www.opcfoundation.org/UA/Part1/

[UA Part 3] OPC UA Specification: Part 3 – Address Space Model
http://www.opcfoundation.org/UA/Part3/

[UA Part 4] OPC UA Specification: Part 4 – Services
http://www.opcfoundation.org/UA/Part4/

[UA Part 5] OPC UA Specification: Part 5 – Information Model
http://www.opcfoundation.org/UA/Part5/

[UA Part 6] OPC UA Specification: Part 6 – Mappings
http://www.opcfoundation.org/UA/Part6/

[UA Part 7] OPC UA Specification: Part 7 – Profiles
http://www.opcfoundation.org/UA/Part7/

[SOAP Part 1] SOAP Version 1.2 Part 1: Messaging Framework
http://www.w3.org/TR/soap12-part1/

[SOAP Part 2] SOAP Version 1.2 Part 2: Adjuncts
http://www.w3.org/TR/soap12-part2/

[XML Encryption] XML Encryption Syntax and Processing
http://www.w3.org/TR/xmlenc-core/

[XML Signature] XML-Signature Syntax and Processing
http://www.w3.org/TR/xmldsig-core/

[WS Security] SOAP Message Security 1.1
http://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf

[WS Addressing] Web Services Addressing (WS-Addressing)
http://www.w3.org/Submission/ws-addressing/

[WS Trust] Web Services Trust Language (WS-Trust)

http://specs.xmlsoap.org/ws/2005/02/trust/WS-Trust.pdf

[WS Secure Conversation] Web Services Secure Conversation Language (WS-SecureConversation)
http://specs.xmlsoap.org/ws/2005/02/sc/WS-SecureConversation.pdf

[SSL/TLS] RFC 2246: The TLS Protocol Version 1.0
http://www.ietf.org/rfc/rfc2246.txt

[X509] X.509 Public Key Certificate Infrastructure
http://www.itu.int/rec/T-REC-X.509-200003-I/e

[HTTP] RFC 2616: Hypertext Transfer Protocol - HTTP/1.1
http://www.ietf.org/rfc/rfc2616.txt

[HTTPS] RFC 2818: HTTP Over TLS
http://www.ietf.org/rfc/rfc2818.txt

[IS Glossary] Internet Security Glossary
http://www.ietf.org/rfc/rfc2828.txt

[NIST 800-12] Introduction to Computer Security
http://csrc.nist.gov/publications/nistpubs/800-12/

[NERC CIP] CIP 002-1 through CIP 009-1, by North-American Electric Reliability Council

[IEC 62351] Data and Communications Security

[SPP-ICS] System Protection Profile – Industrial Control System, by Process Control Security Requirements Forum (PCSRF)

It is assumed that the reader is familiar with Web Services and XML/SOAP. Information on these technologies can be found in [SOAP Part 1] and [SOAP Part 2].

3 Terms, definitions, and abbreviations

3.1 OPC UA Part 1 terms

The following terms defined in [UA Part 1] apply.

1) certificate
2) command
3) event
4) message
5) notification
6) profile
7) subscription

3.2 OPC UA Security terms

3.2.1 Authentication

Authentication is the process of verifying the identity of an entity such as a client, server, or user.

3.2.2 Authorization

An *authorization* is a right or a permission that is granted to a system entity to access a system resource.

3.2.3 Confidentiality

Confidentiality is the protection of data from being read by unintended parties.

3.2.4 Integrity

Integrity is the process of assuring that a message is received as sent, i.e. that it was not modified in transit.

3.2.5 Auditing

Auditing is the tracking of actions and activities in the system, including security related activities. Audit records can be used to verify the operation of system security.

3.2.6 Non-Repudiation

Non-repudiation prevents either the sender or the receiver from denying that it sent or received a transmitted message.

3.2.7 Availability

Availability is the running of the system with unimpeded capacity.

3.2.8 Cyber Security Management System (CSMS)

A *CSMS* is a program designed by an organization to maintain the security of the entire organization's assets to an established level of *confidentiality*, *integrity*, and *availability*, whether they are on the business side or the industrial automation and control systems side of the organization

3.2.9 OPC UA Application

An *OPC UA Application* is an OPC UA *Client*, which calls OPC UA services, or an OPC UA *Server*, which performs those services.

3.2.10 Application Instance

An *Application Instance* is an individual copy, such as an executable, of a program running on one computer. There can be several *Application Instances* of the same application product running at the same time on several computers or possibly the same computer.

3.2.11 Application Instance Certificate

An *Application Instance Certificate* is a *digital certificate* of an individual instance of an application that has been installed in an individual host. Different installations of one software product would have different application certificates.

3.2.12 X.509 Certificate

An X.509 public-key certificate is a public-key certificate in one of the formats defined by X.509 v1, 2, or 3. An [X509] public-key certificate contains a sequence of data items and has a digital signature computed on that sequence.

3.2.13 Digital Certificate

A *digital certificate* is a structure that associates an identity with an entity such as a user or *Application Instance*. The certificate has an associated asymmetric key pair which can be used to authenticate that the entity does, indeed, possess the *private key*.

3.2.14 Security Token

A *security token* is an identifier for a cryptographic key set. All *security tokens* belong to a security context.

3.2.15 Secure Channel

A *secure channel* in OPC UA is a communication path established between an OPC UA client and server that have authenticated each other using certain OPC UA services and for which security parameters have been negotiated and applied.

3.2.16 Nonce

A *nonce* is a random number that is used once and then assigned a different value before the next use. Nonces are often used in algorithms that generate security keys or in a challenge/response test.

3.2.17 Asymmetric Cryptography

Asymmetric cryptography, also known as "public-key cryptography", employs a pair of numeric keys. One of these keys, the *private key*, is known only by the owner of the key pair. The other key, the *public key*, is known by all correspondents of the *private key* owner.

For encryption: In an asymmetric encryption algorithm, such as RSA, when Alice wants to ensure *confidentiality* for data she sends to Bob, she encrypts the data with a *public key* provided by Bob. Only Bob has the matching *private key* that is needed to decrypt the data.

For signature: In an asymmetric digital signature algorithm, such as DSA, when Alice wants to ensure data *integrity* or provide *authentication* for data she sends to Bob, she uses her *private key* to sign the data. To verify the signature, Bob uses the matching *public key* that Alice has provided.

For key agreement: In an asymmetric key agreement algorithm, such as Diffie-Hellman, Alice and Bob each send their own *public key* to the other person. Then each uses their own *private key* and the other's *public key* to compute the new key value. [IS Glossary]

3.2.18 Hash Function

A cryptographic *hash function*, is an algorithm for which it is computationally infeasible to find either a data object that maps to a given hash result (the "one-way" property) or two data objects that map to the same hash result (the "collision-free" property). [IS Glossary]

3.2.19 Public Key Infrastructure (PKI)

A *PKI* is the set of hardware, software, people, policies, and procedures needed to create, manage, store, distribute, and revoke *digital certificates* based on asymmetric cryptography. The core *PKI* functions are to register users and issue their public-key certificates, to revoke certificates when required, and to archive data needed to validate certificates at a much later time. Key pairs for data *confidentiality* may be generated by a certificate authority (CA), but requiring a *private key* owner to generate its own key pair improves security because the *private key* would never be transmitted. [IS Glossary]

3.2.20 Symmetric Cryptography

Symmetric cryptography employs a single “shared” or “secret” key to encrypt messages. *Symmetric cryptography* has a key management disadvantage compared to asymmetric cryptography because of the risk of the key being obtained by an unintended party when it is shared. [IS Glossary]

3.2.21 Message Authentication Code (MAC)

A *MAC* is a short piece of data that results from an algorithm that uses a secret key (see *Symmetric Cryptography*) to hash a message. The receiver of the message can check against alteration of the message by computing a *MAC* that should be identical using the same message and secret key.

3.2.22 Hashed Message Authentication Code (HMAC)

An *HMAC* is a *MAC* that has been generated using an iterative hash algorithm such as MD5 or SHA.

3.2.23 PublicKey

A *public key* is the publicly-disclosable component of a pair of cryptographic keys used for asymmetric cryptography. [IS Glossary]

3.2.24 PrivateKey

A *private key* is the secret component of a pair of cryptographic keys used for asymmetric cryptography.

3.3 Abbreviations and symbols

CSMS	Cyber Security Management System
DSA	Digital Signature Algorithm
DX	Data Exchange
PKI	Public Key Infrastructure
RSA	public key algorithm for signing or encryption, Rivest, Shamir, Adleman
SHA1	Secure Hash Algorithm
SOAP	Simple Object Access Protocol
SSL	Secure Sockets Layer
TLS	Transport Layer Security
UA	Unified Architecture
UML	Unified Modeling Language
WSDL	Web Services Definition Language
XML	Extensible Mark-up Language

3.4 Conventions

3.4.1 Conventions for security model figures

The figures in this document do not use any special common conventions. Any conventions used in a particular figure are explained for that figure.

4 OPC UA Security architecture

4.1 OPC UA Security Environment

OPC-UA is an interface used between components in the operation of an industrial facility at multiple levels: from high-level enterprise management to low-level direct process control of a device. The use of OPC-UA for enterprise management involves dealings with customers and suppliers. It may be an attractive target for industrial espionage or sabotage and may also be exposed to threats through untargeted malware, such as worms, circulating on public networks. Disruption of communications at the process control end causes at least an economic cost to the enterprise and can have employee and public safety consequences or cause environmental damage. This may be an attractive target for those who seek to harm the enterprise or society.

OPC-UA will be deployed in a diverse range of operational environments, with varying assumptions about threats and accessibility, and with a variety of security policies and enforcement regimes. It must, therefore, provide a flexible set of security mechanisms. Figure 1 is a composite that shows a combination of such environments. Some OPC UA clients and servers are on the same host and can be more easily protected from external attack. Some clients and servers are on different hosts in the same operations network and might be protected by the security boundary protections that separate the operations network from external connections. Some *OPC UA Applications* run in relatively open environments where users and applications might be difficult to control. Other applications are embedded in control systems that have no direct electronic connection to external systems.

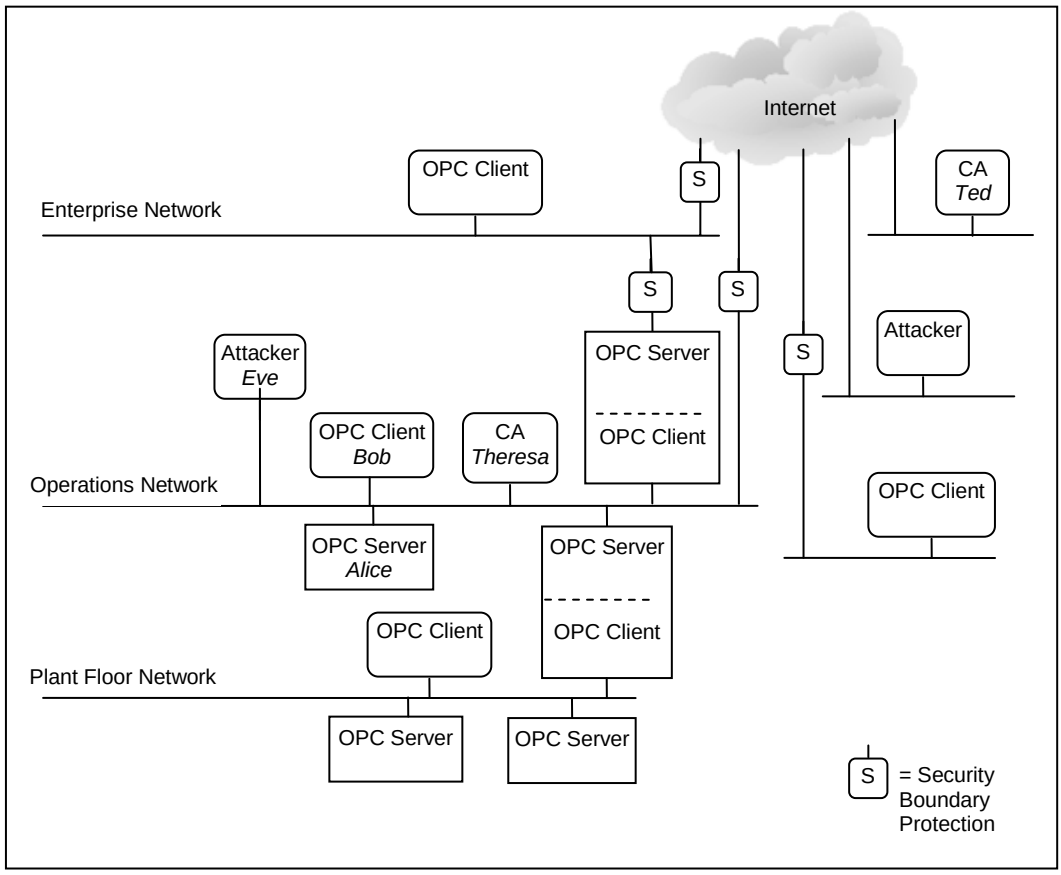

Figure 1 - OPC UA Network Model

OPC UA may run at different places in this wide range of environments. Client and server may communicate within the same host or between hosts within the highly protected control network. Alternatively, OPC UA clients and servers may communicate in the much more open environment over the Internet. The OPC UA activities are protected by whatever security controls the site provides to the parts of the system within which OPC UA runs.

4.2 Security Objectives

4.2.1 Overview

Fundamentally, information system security reduces the damage from attacks. It does this by identifying the threats to the system, identifying the system's vulnerabilities to these threats, and providing countermeasures. The countermeasures reduce vulnerabilities directly, counteract threats, or recover from successful attacks.

Industrial automation system security is achieved by meeting a set of objectives. These objectives have been refined through many years of experience in providing security for information systems in general and they remain quite constant despite the ever-changing set of threats to systems. They are described in the following subsections of Section 4.2. Following the sections that describe the OPC UA security architecture and functions, Section 5.2 reconciles these objectives against the OPC UA functions.

4.2.2 Authentication

Entities such as clients, servers, and users should prove their identities. This *authentication* can be based on something the entity is, has, or knows.

4.2.3 Authorization

The access to read, write, or execute resources should be authorized for only those entities that have a need for that access within the requirements of the system. *Authorization* can be as coarse-grained as allowing or disallowing a client to call a server or it could be much finer grained, such as allowing specific actions on specific information items by specific users.

Authorization must be supported by identification and *authentication* of the parties.

4.2.4 Confidentiality

Data must be protected from passive attacks, such as eavesdropping, whether the data is being transmitted, in memory, or being stored. To provide *confidentiality* data encryption algorithms are used.

4.2.5 Integrity

Receivers must receive the same information that the sender sent, without the data being changed during transmission. Message *integrity* is threatened by message spoofing, message alteration, message replay. Session *integrity* is threatened by session hijacking.

4.2.6 Auditability

The use of the system must be checked to determine that the security measures are effective. Audits are done rigorously in order to provide evidence of secure operation to stakeholders. The system supports *auditing* by recording events that are evidence of security working both well and poorly. These events include new connections, configuration changes, and security error responses to service calls.

4.2.7 Availability

Availability is impaired when the execution of software that needs to run is turned off or when software or the communication system is overwhelmed processing input. Impaired *availability* in OPC UA can appear as slowing down of subscription performance or inability to add sessions for example.

4.3 Security Threats to OPC UA Systems

4.3.1 Overview

OPC UA provides countermeasures to resist the threats to the security of the information that is communicated. The following subsections of Section 4.3 list the currently known threats to environments in which OPC UA will be deployed. Following the sections that describe the OPC UA security architecture and functions, Section 5.1 reconciles these threats against the OPC UA functions.

4.3.2 Message Flooding

An attacker can send a large volume of messages, or a single message that contains a large number of requests, with the goal of overwhelming the OPC server or components on which the OPC server may depend for reliable operation such as CPU, TCP/IP stack, Operating System, or the file system. Flooding attacks can be conducted at multiple layers including OPC-UA, SOAP, [HTTP], or TCP.

Message flooding attacks can use both well-formed and malformed messages. In the first scenario the attacker could be a malicious person using a legitimate client to flood the server with requests. Two cases exist, one in which the client does not have a session with the server and one in which it does.

Message flooding may impair the ability to establish OPC-UA sessions, or terminate an existing session. More generally message flooding may impair the ability to communicate with an OPC-UA entity and result in denial of service.

Message flooding impacts *availability*.

4.3.3 Eavesdropping

Eavesdropping is the unauthorized disclosure of sensitive information that might result directly in a critical security breach or be used in follow-on attacks.

If an attacker has compromised the underlying operating system or the network infrastructure, the attacker might record and capture messages. It may be beyond the capability of a client or server to recover from a compromise of the operating system.

Eavesdropping impacts *confidentiality* directly and threatens all of the other security objectives indirectly.

4.3.4 Message Spoofing

An attacker may forge messages from the client or server. Spoofing may occur at multiple layers in the in the protocol stack. This threat does not include taking over a session, which is described in section 4.3.9 as “Session Hijacking”.

By spoofing messages from the client or server, attackers may perform unauthorized operations and avoid detection of their activities.

Message spoofing impacts *integrity* and *authorization*.

4.3.5 Message Alteration

Network traffic and application layer messages may be captured, modified, and the modified message sent forward to OPC clients and servers. Message alteration allows illegitimate access to a system.

Message alteration impacts *integrity* and *authorization*.

4.3.6 Message Replay

Network traffic and valid application layer messages may be captured and resent to OPC clients and servers at a later stage without modification. An attacker could misinform the user or send in improper command such a command to open a valve but at an improper time.

Message replay impacts *integrity* and *authorization*.

4.3.7 Malformed Messages

An attacker can craft a variety of messages with invalid message structure (malformed XML, SOAP, malicious binary encodings, etc.) or data values and send them to OPC-UA clients or servers.

The OPC client or server may incorrectly handle certain malformed messages by performing unauthorized operations or disclosing unnecessary information. It might result in a denial or degradation of service including termination of the application or, in the case of embedded devices, a complete crash.

Malformed messages impact *integrity* and *availability*.

4.3.8 Server Profiling

An attacker tries to deduce the identity, type, software version, or vendor of the server or client in order to apply knowledge about specific vulnerabilities of that product to mount a more intrusive or damaging attack. The attacker might profile the target by sending valid or invalid formatted messages to the target and try to recognize the type of target by the pattern of its normal and error responses.

Server profiling impacts all of the objectives.

4.3.9 Session Hijacking

An attacker injects valid formatted OPC-UA messages into an existing session by taking over a session.

An attacker may gain unauthorized access to data or perform unauthorized operations.

Session hijacking impacts all of the objectives.

4.3.10 Rogue Server

An attacker builds a malicious OPC-UA server or installs an unauthorized instance of a genuine OPC-UA server.

The OPC client may disclose unnecessary information.

A rogue server impacts all of the security objectives.

4.3.11 Compromising User Credentials

An attacker obtains user credentials such as usernames, passwords, certificates, or keys by observing them on papers, on screens, or in electronic communications; by cracking them through guessing or the use of automated tools such as password crackers.

An unauthorized user could launch and access the system to obtain all information and make control and data changes that harm plant operation or information. Once compromised credentials are used, subsequent activities may all appear legitimate.

Compromised user credentials impact *authorization*.

4.4 OPC UA Relationship to Site Security

OPC UA security works within the overall *cyber security management system* (*CSMS*) of the site. Sites often have a *CSMS* that addresses security policy and procedures, personnel, responsibilities, audits, and physical security. A *CSMS* typically addresses threats that include those that were described in Section 4.3. They also analyze the security risks and determine what security controls the site needs.

Resulting security controls commonly implement a “defense-in-depth” strategy that provides multiple layers of protection and recognizes that no single layer can protect against all attacks. Boundary protections, shown as abstract examples in Figure 1, may include firewalls, intrusion detection systems, controls on dial-in connections, and controls on media and computers that are brought into the system. Protections in components of the system may include hardened configuration of the operating systems, security patch management, anti-virus programs, and not allowing email in the control network. Standards that may be followed by a site include CIP 002-1 through CIP 009-1 and IEC 62351 which are referenced in Section 2.

The security requirements of a site *CSMS* apply to its OPC UA interfaces. That is, the security requirements of the OPC UA interfaces that are deployed at a site are specified by the site, not by the OPC UA specification. OPC UA specifies features that are intended so that conformant client and server products can meet the security requirements that are expected to be made by sites where they will be deployed. Those who are responsible for the security at the site should determine how to meet the site requirements with OPC UA conformant products.

The system owner that installs OPC UA clients or servers must analyze its security risks and must provide appropriate mechanisms to mitigate those risks to achieve an acceptable level of security. OPC UA must meet the wide variety of security needs that might result from such individual analyses. OPC UA clients and servers are required to be implemented with certain security features, which are available for the system owner’s optional use. Each system owner should be able to tailor a security solution that meets its security and economic requirements using a combination of mechanisms available within the OPC UA specification and external to OPC UA.

The security requirements placed on the OPC UA clients and servers deployed at a site are specified by the site *CSMS*, not by the OPC UA specification. The OPC UA security specifications, however, are requirements placed upon OPC UA client and server products, and recommendations of how OPC UA should be deployed at a site in order to meet the security requirements that are anticipated to be specified at the site.

OPC UA addresses some threats as described in the section 4.3. The OPC foundation recommends that vendors address the remaining threats, as detailed in section 6. Threats to infrastructure components that might result in the compromise of client and server operating systems are not addressed by OPC-UA.

4.5 OPC UA Security Architecture

The OPC UA security architecture is a generic solution that allows implementation of the required security features at various places in the OPC UA architecture. Depending on the different mappings described in [UA Part 6], the security functionalities are addressed at different levels. The OPC UA Security Architecture is structured in an Application Layer and a Communication Layer atop the Transport Layer as shown in Figure 2.

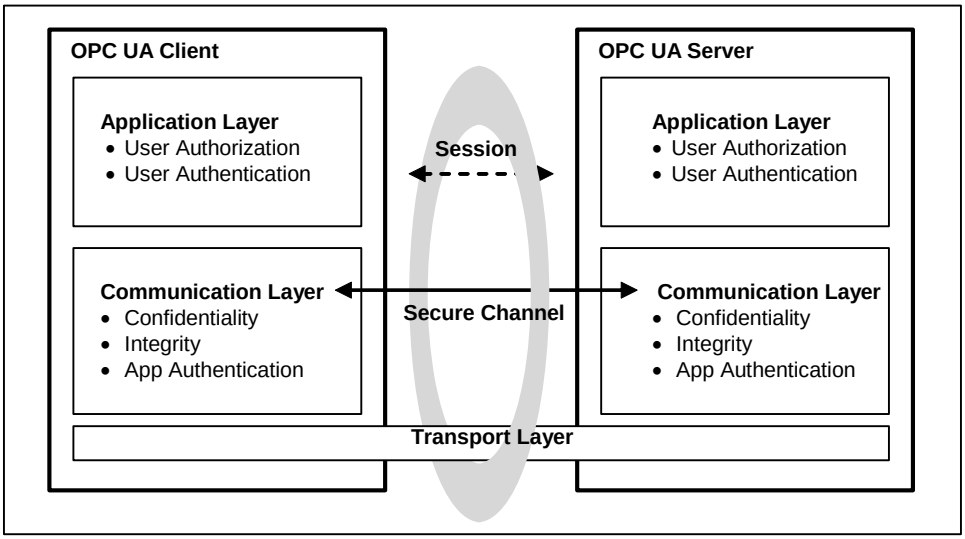

Figure 2 – OPC UA Security Architecture

The routine work of a client application and a server application to transmit plant information, settings, and commands is done in a session in the application layer. The application layer also manages the security functions of user *authentication* and user *authorization*. The security functions that are managed by the Application Layer are provided by the Session Services that are specified in [UA Part 4]. A session in the Application Layer communicates over a *secure channel* that is created in the Communication Layer and relies upon it for secure communication. All of the session data is passed to the Communication Layer for further processing.

Although a session communicates over a *secure channel*, the binding of users, sessions, and *secure channels* is flexible. Impersonation allows the user of the session to change. A session can have a different user than the user that created the *secure channel*. To survive the loss of the original channel and resume with another, the implementation of the communication channel is responsible to re-establish the connection without interrupting the logical secure channel.

The Communication Layer provides security functionalities to meet *confidentiality*, *integrity* and application *authentication* as security objectives. The provided security functionalities together with negotiated and secret information are used to establish a *secure channel* between a client and a server. This logical channel provides encryption to maintain *confidentiality*, signatures to maintain *integrity* and certificates to provide application *authentication* for data that comes from the Application Layer and passes the “secured” data to the Transport Layer. The security functions that are managed by the Communication Layer are provided by the Secure Channel Services that are specified in [UA Part 4].

The security functions provided by the *secure channel* services are implemented by a protocol stack that is chosen for the implementation. Mappings of the services to some of the protocol stack options are specified in [UA Part 6] which details how the functions of the protocol stack are used to meet the OPC UA security objectives.

The Communication Layer can represent an OPC UA protocol stack. OPC UA specifies two alternative stack mappings that can be used as the Communication Layer. These mappings are UA Native mapping and Web Services mapping.

If the UA Native mapping is used, then functionalities for confidentiality, integrity, application authentication, and the secure channel are similar to the [TLS/SSL] specifications, as described in detail in [UA Part 6].

If the Web Services mapping is used, then [WS Security], [WS Secure Conversation] and [XML Encryption] are used to implement the functionalities for confidentiality, integrity, application authentication as well as for implementing a *secure channel*. For more specific information, see [UA Part 6].

The Transport Layer handles the transmission, reception and the transport of data that is provided by the Communication Layer.

4.6 Policy

Security policies specify which security mechanisms are to be used. Security policies are used by the server to announce what mechanisms it supports and by the client to select one of those available policies to be used for the secure channel it wishes to open. The mechanisms specified include the following:

- which messages will be signed: choice of: all messages or no messages at all
- whether to encrypt all messages or no messages
- algorithms for signing and encryption

The choice of policy is normally made by the control system administrator, typically when the client and server products are installed.

If a server serves multiple clients, it maintains separate policy selections for the different clients. This allows a new client to select policies independent of the policy choices that other clients have selected for their secure channels.

[UA Part 7] specifies several policies. To improve interoperability among vendors' products, server products should implement these policies rather than define their own.

The security policy structure may be extended in later OPC UA versions, so the security policy structure allows additional fields so that it can be compatible with servers that conform to later versions of OPC UA.

4.7 Profile

OPC UA client and server products are certified against profiles that are defined in [UA Part 7]. Some of the profiles specify security functions and others specify other functionality that is not related to security. The profiles impose requirements on the certified products but they do not impose requirements on how the products are used. A consistent minimum level of security is required by the various profiles. However, different profiles specify different details, such as which encryption algorithms are required for which UA functions. If a problem is found in one encryption algorithm, then the OPC foundation can define a new profile that is similar, but that specifies a different encryption algorithm that does not have a known problem. [UA Part 7], not this Part 2, is the normative specification of the profiles.

Policies refer to many of the same security choices as profiles, however the policy specifies which of those choices to use in the session. The policy does not specify the range of choices that the product must offer like the profile does.

Each security mechanism in OPC UA is provided in client and server products in accordance with the profiles with which the client or server complies. At the site, however, the security mechanisms

may be deployed optionally. In this way each individual site has all of the OPC UA security functions available and can choose which of them to use to meet its security requirements.

4.8 User Authorization

UA provides a mechanism to exchange user credentials but does not specify how the applications use these credentials. Client and server applications may determine in their own way what data is accessible and what operations are authorized.

4.9 User Authentication

User *authentication* is provided by the Session Services with which the client passes user credentials to the server as specified in [UA Part 4]. The client and server can each authenticate the user with these credentials.

The user who is communicating over a session can be changed using the ImpersonateUser service in order to meet needs of the application. This does not change the identity of the user who created the *secure channel* over which the session is currently communicating.

4.10 OPC UA Security Services

A set of cryptographic keys is used to encrypt and sign messages in order to protect *confidentiality* and *integrity*. They are negotiated by services of the *secure channel* service set. Session keys are changed periodically so that attackers do not have unlimited time and unrestricted sequences of messages to use to determine what the keys are. The functions for managing these keys are specified in [UA Part 4].

4.11 Auditing

Clients and servers generate audit records of successful and unsuccessful connection attempts, results of security option negotiations, configuration changes, system changes, and session rejections.

OPC UA provides support for security audit trails through two mechanisms. First, it provides for traceability between client and server audit logs. The client generates an audit log entry for an operation that includes a request. When the client issues a service request, it generates an audit log entry and includes the local identifier of the log entry in the request sent to the server. The server logs requests that it receives and includes the client's entry id in its audit log entry. In this fashion, if a security-related problem is detected at the server, the associated client audit log entry can be located and examined. OPC UA does not require the audit entries to be written to disk, but it does require that they be available. OPC UA provides the capability for servers to generate event notifications that report auditable events to clients capable of processing and logging them.

Second, OPC UA defines standard audit parameters to be included in audit records. This promotes consistency across audit logs and in *audit* events. [UA Part 5] defines the data types for these parameters. [UA Part 7] defines profiles, which include the ability to generate audit events and use these parameters, including the client audit record id.

The following clauses illustrate the behavior of OPC-UA servers and clients that support *auditing*.

4.11.1 Single client and server

Figure 3 illustrates the simple case of a client communicating with a server.

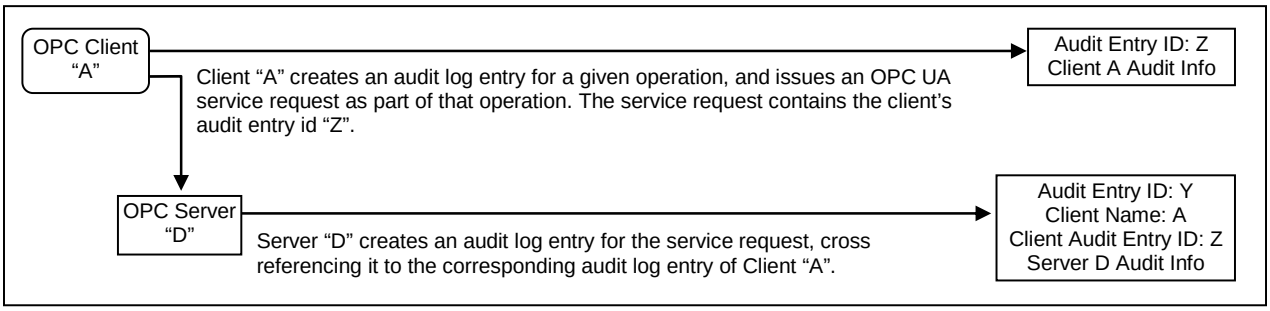

Figure 3 – Simple Servers

In this case, Client “A” executes some auditable operation that includes the invocation of an OPC UA service in Server “D”. It writes its own audit log entry, and includes the identifier of that entry in the service request that it submits to the server.

The server receives the request and creates its own audit log entry for it. This entry is identified by its own audit id and contains its own *auditing* information. It also includes the name of the client that issued the service request and the client audit entry id received in the request.

Using this information, an auditor can inspect the collection of log entries of the server and relate them back to their associated client entries.

4.11.2 Aggregating server

Figure 4 illustrates the case of a client accessing services from an aggregating server. An aggregating server is a server that provides its services by accessing services of other OPC UA servers, referred to as lower layer-servers.

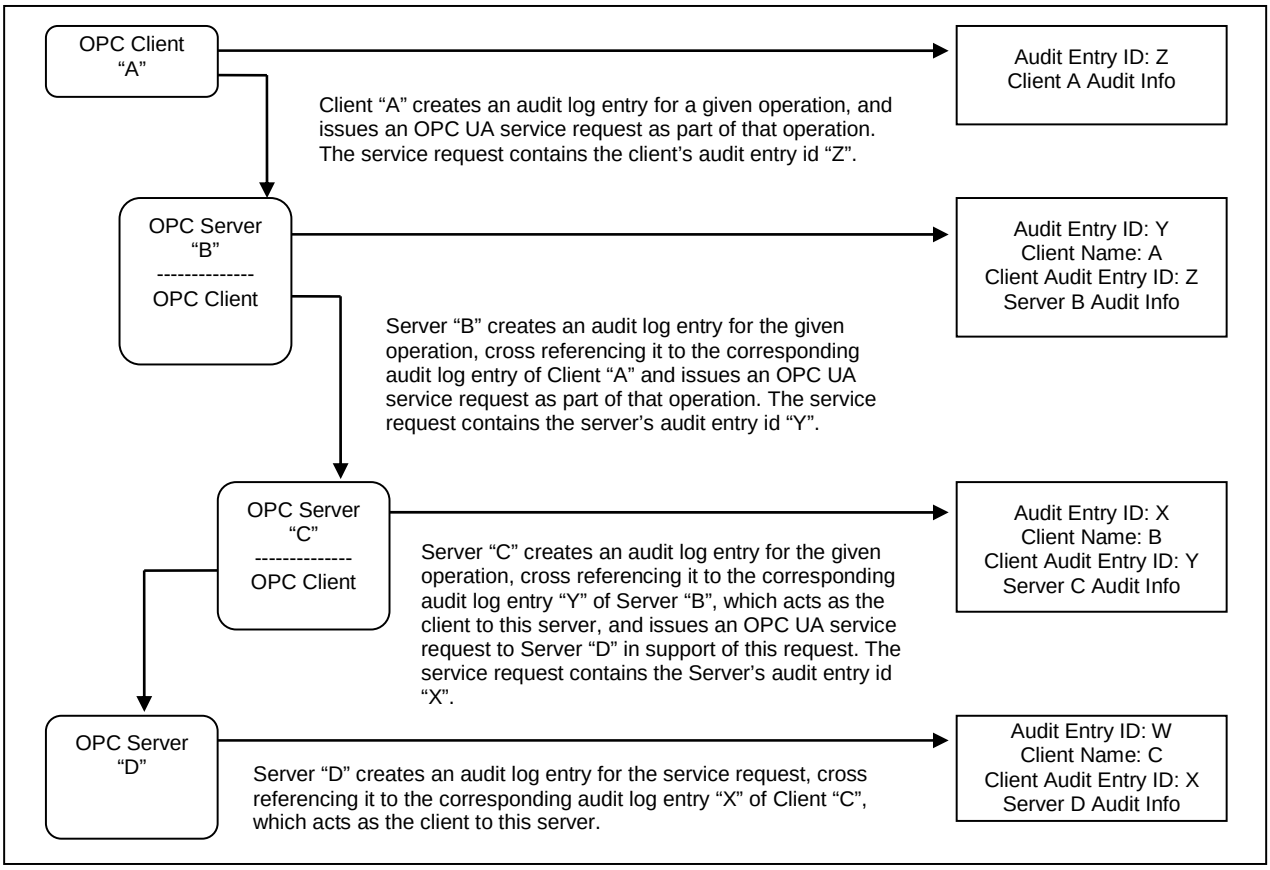

Figure 4 – Aggregating Servers

In this case, each of the servers receives requests and creates its own audit log entry for them. Each entry is identified by its own audit id and contains its own *auditing* information. It also includes the name of the client that issued the service request and the client audit entry id received in the request. The server then passes the audit id of the entry it just created to the next server in the chain.

Using this information, an auditor can inspect the server's log entries and relate them back to their associated client entries.

In most cases the servers will only generate Audit events, but these Audit events will still contain the same information as the Audit log records. In the case of aggregating servers a server would also be required to subscribe for audit events from the servers it is aggregating. In this manner, Server "B" would be able to provide all of the audit events to Client "A", including the event generated by Server "C" and server "D"

4.11.3 Aggregation through a non-auditing server

Figure 5 illustrates the case of a client accessing services from an aggregating server that does not support *auditing*.

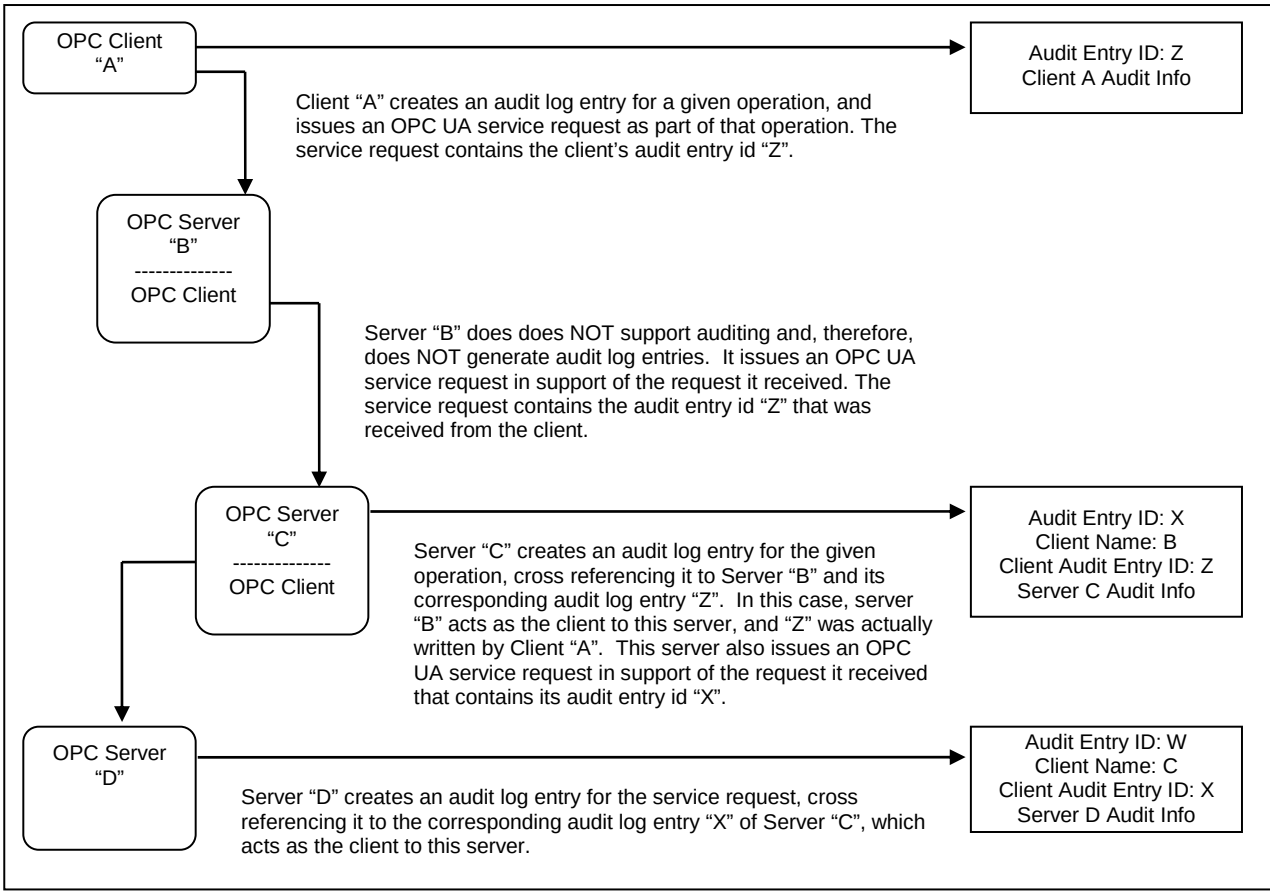

Figure 5 – Aggregation with a Non-Auditing Server

In this case, each of the servers receives their requests and creates their own audit log entry for them, with the exception of Server "B", which does not support *auditing*. In this case, Server "B" passes the audit id it receives from its Client "A" to the next server. This creates the required audit chain. Server "B" is not listed as supporting *auditing*. In a case where a server does not support writing audit entries, the entire system may be considered as not supporting *auditing*.

In the case of an aggregating server that does not support *auditing*, the server would still be required to subscribe for audit events from the servers it is aggregating. In this manner, Server "B" would be able to provide all of the audit events to Client "A", including the event generated by Server "C" and server "D", even though it did not generate an audit event.

4.11.4 Aggregating server with service distribution

Figure 6 illustrates the case of a client that submits a service request to an aggregating server, and the aggregating service supports that service by submitting multiple service requests to its underlying servers.

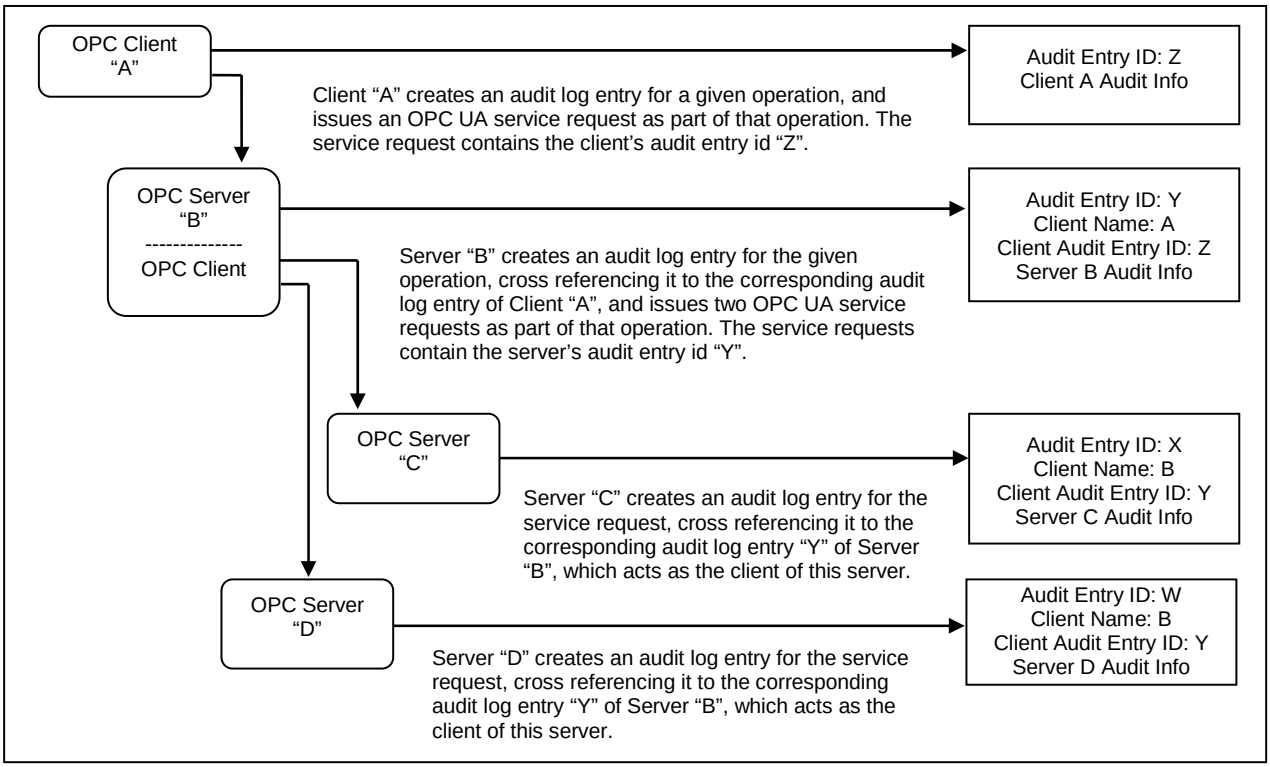

Figure 6 – Aggregate server with Service Distribution

In the case of aggregating servers a server would also be required to subscribe for audit events from the servers it is aggregating. In this manner, Server “B” would be able to provide all of the audit events to Client “A”, including the event generated by Server “C” and server “D”

5 Security Reconciliation

5.1 Reconciliation of Threats with OPC UA Functions

5.1.1 Overview

The following subsections of Section 5.1 reconcile the threats that were described in Section 4.3 against the OPC UA functions. Each of the following subsections relates directly to the threat described in its corresponding subsection of Section 4.3.

Compared to the reconciliation with the objectives that will be given in section 5.2, this is a more specific reconciliation that relates OPC UA security functions to specific threats.

5.1.2 Message Flooding

OPC UA minimizes the loss of *availability* caused by message flooding by minimizing the amount of processing that must be done with a message before the message is authenticated. This prevents an attacker from leveraging a small amount of effort to cause the legitimate *OPC UA Application* to spend a large amount of time responding, thus taking away processing resources from legitimate activities. GetSecurityPolicies (specified in [UA Part 4]) and OpenSecureChannel (specified in [UA Part 4]) are the only services that the server must handle before the client is recognized. The response to GetSecurityPolicies is only a small set of static information so the server does not need to do much processing. The response to OpenSecureChannel consumes significant server resources because of the signature and encryption processing. The server implementation could protect itself from floods of OpenSecureChannel messages in two ways: first, the server could intentionally delay its processing of OpenSecureChannel requests once it receives a bad one and issuing an alarm would alert plant personnel that an attack is underway that could still be blocking new legitimate OpenSecureChannel calls; second, when an OpenSecureChannel request attempts to exceed the server's specified maximum number of concurrent channels the server must give an error response without performing the signature and encryption processing. Certified OPC UA servers are required to specify their maximum number of concurrent channels in their product documentation as specified in [UA Part 7].

OPC UA user and client *authentication* reduce the risk of a legitimate client being used to mount a flooding attack. See the reconciliation of *authentication* in Section 5.2.2.

OPC UA *auditing* functionality provides the site with evidence that can help the site discover that flooding attacks are being mounted and find ways to prevent similar future attacks. See Section 4.11.

OPC UA relies upon the site *CSMS* to prevent attacks such as message flooding at protocol layers and systems that support OPC UA.

5.1.3 Eavesdropping

OPC UA provides encryption to protect against eavesdropping as described in Section 5.2.4 - Confidentiality.

5.1.4 Message Spoofing

As specified in the OpenSecureChannel service in [UA Part 4], OPC UA counters message spoofing threats by the possibility to sign messages. Additionally all messages must contain a valid session id and are assigned to a *secure channel*.

5.1.5 Message Alteration

OPC UA counters message alteration by the signing of messages that is specified in the OpenSecureChannel service specified in [UA Part 4]. If messages are altered, checking the signature will reveal any changes and allow the recipient to discard the message.

5.1.6 Message Replay

OPC UA uses session ID, timestamps, and sequence numbers for every message request and response. If messages are additionally signed then messages cannot be replayed without detection. Timestamps are specified in the RequestHeader section in [UA Part 4]. Signing is specified in the OpenSecureChannel service in [UA Part 4]. Sequence numbers are specified in the section that specifies the Write, Call, and Notification Message services in [UA Part 4].

5.1.7 Malformed Messages

Implementations of OPC UA client and server products counter threats of malformed messages by checking that messages have the proper form and that parameters of messages are within their legal range. This is specified in [UA Part 6] and a similar specification regarding range checking of parameters of services is in [UA Part 4].

5.1.8 Server Profiling

OPC UA limits the amount of information that servers provide to clients that have not yet been identified. This information is the response to the GetServerPolicies service specified in [UA Part 4].

Compliance testing might improve the consistency of error handling by OPC UA products from different vendors.

5.1.9 Session Hijacking

OPC UA counters session hijacking by assigning a security context with each session as specified in the CreateSession service in [UA Part 4]. Hijacking a session would thus first require compromising the security context.

5.1.10 Rogue Server

OPC UA counters the use of rogue servers to collect information from a client by requiring servers to authenticate to clients as specified in the OpenSecureChannel service in [UA Part 4].

5.1.11 Compromising User Credentials

OPC UA protects user credentials sent over the network by encryption as described in Section 5.2.4 - Confidentiality.

OPC UA depends upon the site *CSMS* to protect against other attacks to gain user credentials, such as password guessing or social engineering.

5.2 Reconciliation of Objectives with OPC UA Functions

5.2.1 Overview

The following subsections of Section 5.2 reconcile the objectives that were described in Section 4.2 with the OPC UA functions. Each of the following subsections relates directly to the objective described in its corresponding subsection of Section 4.2.

Compared to the reconciliation against the threats of section 5.1, this reconciliation justifies the completeness of the OPC UA security architecture.

5.2.2 Authentication

OPC UA Applications support *authentication* of the entities with which they are communicating as well as providing the necessary *authentication* credentials to the other entities.

Application Authentication

As specified in the OpenSecureChannel service in [UA Part 4], OPC UA client and server applications identify and authenticate themselves with [X509] certificates. Some choices of the Communication Stack require these certificates to represent the machine or user instead of the application.

User Authentication

As described in the OpenSecureChannel service in [UA Part 4], the OPC UA client accepts a user identity token from the user, authenticates the token, and passes it to the OPC UA server. The OPC UA server authenticates the user token. *OPC UA Applications* accept tokens in any of the following three forms: username/password, [X509] v3, or Web Services compliant. Clients and servers accept username/password tokens.

As specified in the sections of the CreateSession and ActivateSession services in [UA Part 4], the user identity token is validated with a challenge-response process using the ClientNonce and ServerNonce. The server provides a *nonce* and signing algorithm as the challenge in its CreateSession response. The client responds to the challenge by signing the server's *nonce* and providing it as an argument in its subsequent ActivateSession call.

5.2.3 Authorization

OPC UA does not specify how user or client *authorization* is to be provided. *OPC UA Applications* that are part of a larger industrial automation product may manage *authorizations* consistent with the *authorization* management of that product. Identification and *authentication* of users is specified in OPC UA so that client and server applications can recognize the user in order to determine the *authorizations* of the user.

OPC UA servers respond with the Bad_UserAccessDenied error code to indicate an *authorization* error as specified in the status codes section of [UA Part 4].

5.2.4 Confidentiality

OPC UA provides encryption to protect *confidentiality*. Encryption is specified in [UA art 6] in the sections on encryption and decryption in the UA Native Mapping and in the section on encrypting messages of [WS Security] in the XML Web Services Mapping.

OPC UA relies upon the site *CSMS* to protect *confidentiality* on the network and system infrastructure. OPC UA relies upon the *CSMS* to manage keys.

5.2.5 Integrity

Integrity is threatened by message spoofing, message alteration, message replay, malformed messages, session hijacking which are reconciled with OPC UA countermeasures in their respective subsections of Section 5.1. The signing with changing keys of messages that contain session ID, timestamp, and sequence number provides a complete foundation to protect *integrity*.

5.2.6 Auditability

As specified in the UA Auditing section of [UA Part 4], OPC UA supports audit logging by providing traceability of activities through the log entries of the multiple clients and servers that initiate, forward, and handle the activity. OPC UA depends upon *OPC UA Application* products to provide an effective audit logging scheme or an efficient manner of collecting the Audit events of all nodes, This scheme may be part of a larger industrial automation product of which the *OPC UA Applications* are part.

5.2.7 Availability

OPC UA minimizes the impact of message flooding as described in Section 5.1.2.

Some attacks on *availability* involve opening more sessions than a server can handle thereby causing the server to fail or operate poorly. Servers reject sessions that exceed their specified maximum number.

6 Implementation considerations

This section provides guidance to vendors that implement OPC-UA applications. Since many of the countermeasures required to address the threats described above fall outside the scope of the OPC-UA specification, the advice in this section suggests how some of those countermeasures should be provided.

For each of the following areas, this section defines the problem space, identifies consequences if appropriate countermeasures are not implemented and recommends best practices.

6.1 Application Instance Certificates

It is recommended that all UA applications create a unique, self-signed root certificate which is defined as the *Application Instance Certificate* whenever a new instance of the application is installed. The *public key* for this auto-generated certificate should be stored in a location accessible to the administrator. This will allow the administrator to configure other UA applications that need to communicate with the new UA *Application Instance*. The UA application should also allow the administrator to replace the auto-generated certificate with one issued by a CA selected by the administrator. The trusted certificate list should be protected at the site from unauthorized alteration.

It also recommended that UA applications simplify the process of adding new self-signed certificates to its list of trusted certificates. If possible, the UA application should generate some sort of notification whenever UA applications exchange certificates using UA services for the first time. The best process depends on the application. However, it could include displaying a dialog to the user and asking the user to confirm that the certificate is acceptable. Server applications or applications without a user interface could conditionally accept the certificate and report it to an administrator via a security log or e-mail. UA applications should never silently accept certificates that are either self-signed or signed by an unknown certification authority.

6.2 Appropriate Timeouts:

Time outs, the time that the implementation must wait (usually for an event such as message arrival), play a very significant role in influencing the security of an implementation. Potential consequences include

- Denial of service: Denial of service conditions may exist when a client does not reset a session, if the timeouts are very large.

- Resource consumption: When a client is idle for long periods of time, the server must keep the client information for that period, leading to resource exhaustion.

The implementer should use reasonable timeouts for each connection stage.

6.3 Strict Message Processing

The specifications often specify the format of the right messages and are silent on what the implementation should do for messages that deviate from the specification. Typically, the implementations continue to parse such packets, leading to vulnerabilities.

- The implementer should do strict checking of the message format and should either drop the packets or send an error message as described below.

- Error handling shall use the error code, defined in [UA Part 4], that most precisely fits the condition.

6.4 Robust Error recovery

An error (hang, entering wrong protocol state, etc.) in the protocol may occur due to various reasons such as improper specification of the protocol or the client sending messages that are out of the scope of the specification. Graceful error recovery practices should be implemented so that the implementation recovers from the error. Sending the right type of error messages and reclaiming the memory are examples of such practices.

6.5 Random Number Generation

Random numbers that meet security needs can be be generated by suitable functions that are provided by cryptography libraries. Other means are too weak, such as using CRT rand().

6.6 Special and Reserved Packets

The implementation must understand and correctly interpret any message types that are reserved as special (such as broadcast and multicast addresses in IP specification). Failing to understand and interpret those special packets may lead to vulnerabilities.

6.7 Rate Limiting and Flow Control

OPC-UA does not provide rate control mechanisms, however an implementation can incorporate rate control.

7 Site Recommendendations

In later releases of this specification, this section will contain recommendations of actions that should be taken at the site in order to provide security of OPC UA.