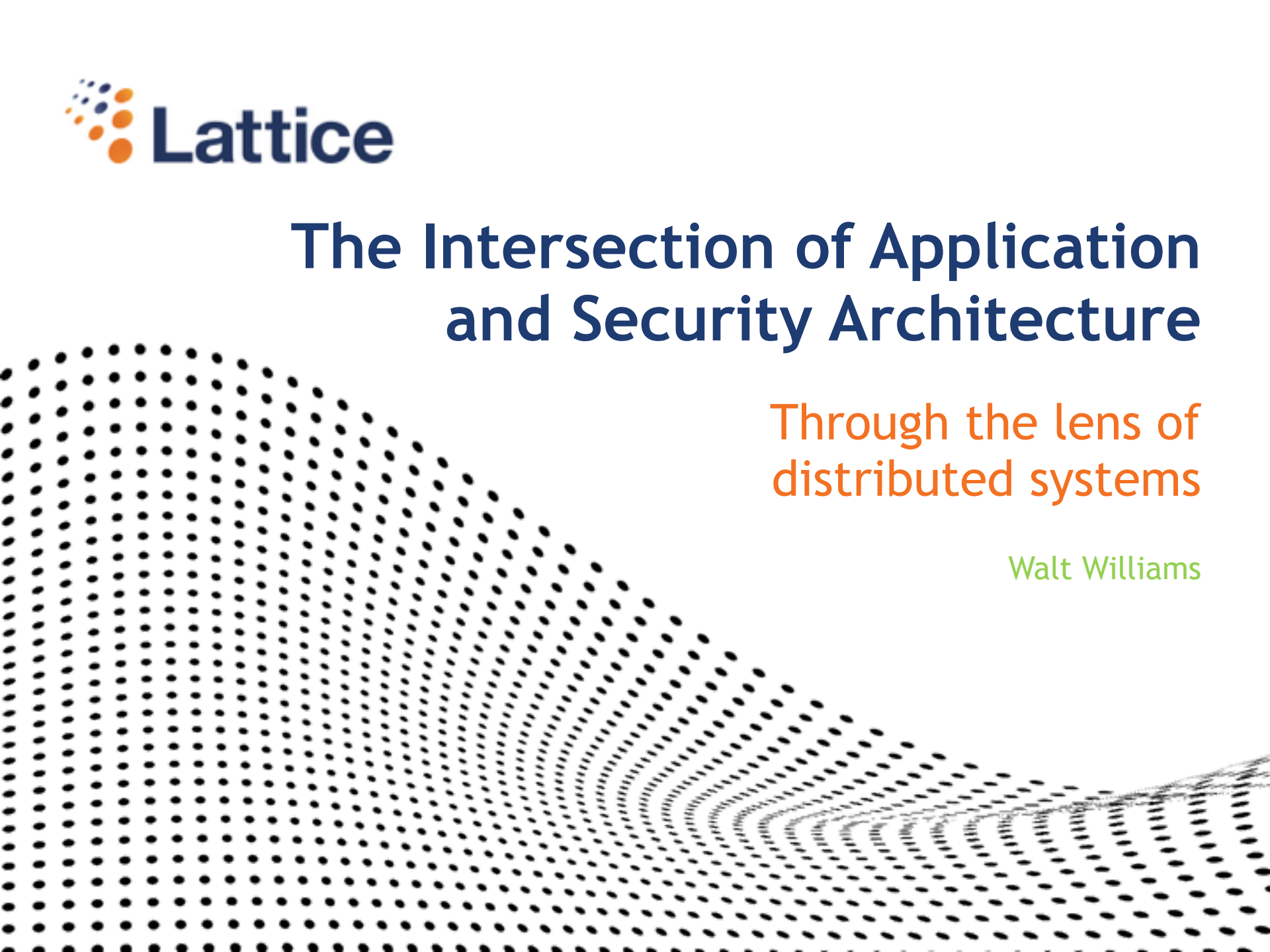


The Intersection of Application and Security Architecture

Through the lens of distributed systems

Walt Williams

Who is this bloke?

- Author of Security for Service Oriented Architecture, CRC press 2014
- Director of Information Security, Lattice Engines
- Has done almost everything in Information Security, from Identity and Access Management, provisioning, single sign-on, penetration testing, incident response, disaster recovery, compliance testing, architecture and design, etc.
- @LESecurity
- wwilliams@lattice-engines.com
- Walt.williams@gmail.com

The Problem within Information Security

- Security architecture may start with policy but most architects don't take it beyond infrastructure
- Firewalls
- Network segregation
- IDS/IPS
- Proxies, etc
- Applications just fit into a layer of the grand scheme of things (in other words, they're an afterthought)
  - Layer 7

The problem within Application Development

- Application architectures start with functionality
- Workflow
- Data access
- Reporting
- The slick new GUI, etc
- Security fits just fine in the grand scheme of things
  - It all ends when the account authenticates

Best baked in during design

- Application architects need to understand security requirements
  - It is more than just authentication and provisioning
- Security architects need to understand what goes into a good and secure application design
  - It is more than just authentication and account management

Applications are like Ogres

- Applications have layers
  - Presentation Layer
  - Data Layer
  - Workflow Layer
  - Messaging Layer
  - Service Layer
  - Business Layer
- Security is part of what should transpire as one layer communicates with another

Some things we all know – or should

- Distributed applications have no perimeters
- Distributed applications may have no single account store
- Distributed applications may have no single centralized log
- Distributed applications will use multiple protocols and frameworks as best fits the functionality in use at that moment
  - Thus one application may use both REST and SOAP

Why bake security in?

- Doesn't DevOps Solve all of this?
  - No
  - With Devops, you do things at speed
- It is an often unarticulated business requirement
  - Unarticulated often because Security is not invited to the requirements meeting
  - The business people just presume the application people get how to design a secure application

If you were invited to the party

- Do you understand application security beyond the OWASP top 10?
  - And will all due respect, the OWASP top 10 is not a good place to start, if you must use an independent third party threat classification system, try CAPEC
  - https://capec.mitre.org/
  - Besides, a threat list is not a design

Lets work from some requirements

- Company A
  - A business to business application which provides review and testing of integrated circuit boards designs for efficiency using mathematical algorithms proprietary to this imaginary company
- Company B
  - A corporation is looking to develop a business to consumer travel and entertainment application that will be leveraging services provided by other providers
- Company C
  - The corporation provides updated commodities prices for a commodities market upon request. This is a very simple service that can be accessed by individual traders or other service providers which can wrap the information we provide into their platform
- Company D
  - A distributed financial application for a large international bank. The Bank has a legacy system on a mainframe that has been providing account management for its customers for decades

Company A

- Will be interfacing with many customers' design teams,
  - These customers, who are often competitors can't access each other's designs
- That the designs are not tampered with.
- They provide to their customer an existing database of chips, capacitors, resistors, etcetera so that each design may be properly tested with the components that will be soldered to the board.
- This means that the application has some data that is common and shared between customers and other data which is proprietary and not shared.
- The customer must maintain control over who has access to their design documents and schematics

Company B

- Maps will be provided by some map company, Weather information by the national weather service,
- What to do by local convention and visitor's bureaus
- Make hotel reservations, train and airplane reservations and book local rental car and or taxi services.
- A key differentiator for this business is that it books mystery trips, where the person doing the booking has no idea where they'll be going until the transaction is complete, but you, as the service provider must be able to keep track and present enough information to allow an intelligent decision that meets the customer's budget and whimsy.
- As a mystery travel service, the names and locations of the hotels, the airline destinations will be kept anonymous until after the booking is complete.
- Private and personal data will be collected as part of the booking process
- Because sometimes life changes and you can't keep the plans you've made, for what ever reason the customer must be able to change or cancel their reservations at any time.
- You will allow various travel agencies to create accounts on your system to that they may make arrangements for their customers
- The arrangements made by these agents must be able to be shared with the consumer.
- Some of your customers will want to share their arrangements with family and friends via social networking sites

Company C

- This is a very simple service that can be accessed by individual traders or other service providers which can wrap the information we provide into their platform.
- We don't accept orders nor can our users use our service to sell.
- We can, and do, however provide historical analysis for any time period since the founding of the exchange we're the front end for.
- We will need to support two data feed models to support our diverse customer base: the push and the pull.
- Because we don't care who accesses our service to pull data from us, there is no need for an identity service provider nor for a authentication front end.
- However, we must prove that the data we represent is ours to the organizations that require us to push the prices to them.
- It is for this reason that we're not supporting WebSockets at this time, as there is no mechanism to demonstrate authenticity of the data.
- However, we are looking at the use of extended validated certificates to see if they meet our risk profile of our customers for sufficiently demonstrating authenticity and integrity of the data so that we can provide an HTML 5 WebSockets interface to our data.

Company D

- Extend the power of a legacy system through the Internet
- The bank has deployed a message bus internally to govern and control all communications with the legacy system
- A subset of the application's functionality needs to be exposed to Bank customers.
- This is done to individual customers through a web portal.
- Corporate customers can access a business centric portal similar in design to the customer portal
- Corporate customers may also access the service through service calls of their own.

Application Architecture

- Key Principles
  - Design to Last
  - Design to Change
- Frameworks
  - J2EE
  - .NET
  - REST
  - SOAP
  - HTML 5

Layers

- Presentation
- Business
- Data
- Workflow
- Communications
- Service

Elements which cross the layers

- Instrumentation
- Authentication
- Authorization
- Exception Management
- Communication*
- Caching
- Logging

Presentation Layer

- Sometimes it is the GUI
- Sometimes it is the API
  - This may include a WSDL to provide instructions for calling the API
- Both point of access and point of attack
- GUI consists of interface components and presentation logic
  - Common mistake is to use presentation logic for security
  - While the authentication and authorization components must be present, the logic that governs their action should be independent of the presentation logic.
  - Logic should concern itself with:
    - Data Caching
    - Communication*
    - Composition
    - Error Handling
    - Navigation

Business Layer

- Not all application architects see the need for this
  - This would place all business logic in either the presentation layer or the data layer
  - From experience this leads to both performance and security issues
  - When business logic is coupled with presentation logic, it is hard to maintain and grow either.
  - Changes to the visual presentation of the application are necessarily more complex and prone to mistakes.
  - Changes to the data model and communication become mired in a need to preserve logic tied to an outdated data model.
  - Application support and maintenance under these conditions eventually deteriorate to the point where application changes become odious and the business either begins to suffer in its reputation with its customers due to an increasingly amount of bugs, or the business halts investment in the application, and all development is relegated to the role of patch/fix.

Data Layer

- The data layer is comprised of both the actual data
- The code used to manipulate and query that data.
- The data layer of an application architecture is NOT the data architecture.
- However, the data architecture must be understood
- May be static or dynamic in nature.
  - Where static, such as with a database stored procedure, there is the opportunity to optimize the language for efficiency and accuracy.
  - The problem is that this depends upon you knowing in advance and planning for all possibilities, and that is not always possible.
  - Security geeks will tout static (stored procedures) as the savior for your data layer woes and pout much if you don't do this
  - The solution is actually in the communications layer

Workflow

- An application's workflow is the logic through which events happen in a desired sequence.
- Frequently planned out in a logic tree or flow chart, it may be as simple as select product, place into shopping cart, check out, pay for purchase and receive confirmation.
- The trouble is that if you mandate this workflow, than none of your customers can ever order more than one product per order, a strategy designed to quickly drive a company bankrupt.
- There are three basic kinds of workflow
  - Simple sequential workflow.
    - Sequential workflows may have conditional branching
    - All events must follow a clearly defined sequence.
    - While not appropriate for an e-business application which is consumer facing, it may be very appropriate for an event creation and management solution.
  - Machine State Workflow
    - For applications that depend upon a machine state, where a stage of the application must be completed before the next phase must begin.
    - This kind of workflow is appropriate for applications which spin up more instances when CPU load is getting high etc.
  - Data state Workflow
    - Can be leveraged by applications that are more transitionally oriented

Communications

- This layer is also hotly debated
- There ARE benefits from integrating communications into each layer
  - Even if you have a communications layer, each layer must know how to call it
  - And how to act on calls from it
- This could be a message oriented
  - Message based communications are often used when crossing physical or process boundaries due to their ability to be queued in case of system failure or network interruption. Message queues can support transactional systems and allow for reliable once only delivery.
- This could be object oriented
  - Object based communications are often used when only logical boundaries need to be crossed, such as between tiers of a multi tiered application which is not distributed across corporations, however, they can be used to great effect in a service oriented architecture due to their ability to encapsulate information.
- This could be direct method calls
  - This technique limits your ability to distribute your application across business perimiters

Some details on messages

- Messaging systems often have different components, and can represent a mini-architecture within the application architecture. They may contain channel adapters, which are components, which can access the API and application data directly, publishing messages based upon the data and invoking functionality through API calls.
- Many organizations use a message bus internally to link together disparate systems which use divergent protocol sets
- Duplex messages are two way communications sent independently.
- Fire and Forget are messages sent with no reply required or expected.
- Reliable Sessions are messages sent end to end regardless of the intermediaries.
- Request/Response are like duplex messages in so far as they are messages between two components, but each message requires a distinct and dependent response.

Service Layer

- Could be described as a slow asynchronous messaging system which exposes an API as a presentation layer for distributed use of application functionality
- Should be scoped to deliver an application's functionality
- Designed as customer agnostic
  - With the broadest representation of the application functionality made available.
- Should not expose other layers of the application
- Should be separated from the infrastructure as much as possible.
- You can use REST or SOAP or other protocols as the method to call services
- Should be able to detect and manage invalid requests
- Should be able to detect and manage repeated messages
- Unlike a message layer, a service layer can be in and of itself transformative
- In many ways it resembles the presentation layer, business layer, communications layer and data layer

More on Services

- Like a Presentation Layer, Service Layers must be concerned with:
  - Authentication & Authorization
  - Communication
  - Exception Management
- However a service must also manage many of the same concerns as the messaging layer of the application:
  - Message Construction
  - Message Endpoint
  - Message Protection
  - Message Routing
  - Message Transformation
  - Message Validation

Now for a few words on Service Oriented Architectures

- Service-Oriented Architecture was developed to solve the problem of how to connect disparate systems in a way that they could function together in a systematic manner. These systems, or applications each provide to the organization the value for which they are designated, but often this value would be enhanced if either the processing or the results could be integrated with the information processing of other applications.

Service Oriented Architectures

- Are the ultimate expression of architecture as application
  - And unlike an internally facing single application, it can only really work well if security is baked in at all layers
  - The lessons learned from building service oriented architectures can help build any distributed application
  - Traditionally they are layered as follows:
    - Application Frontend (Presentation Layer)
    - Service (Service Layer)
    - Service Repository (Data Layer)
    - Service Bus (Communications Layer)

Kinds of services

- Utility Service
- Business Service
- Controller Service
- Proxy Service
- Wrapper Service
- Coordination Service
- Process Service

What about Security?

- Security has layers too
- Security architectures start with risk management through policy
- Address issues of
  - Confidentiality
  - Availability
  - Integrity
  - Control
  - Utility
  - Authenticity
  - Privacy

Fundamental Principles

- Least Privilege
  - Accounts are provided with only what they need to execute their responsibilities
- Least Authority
  - Applications run under the minimum authority to provide business functionality
- Separation of Duties
  - Clear conflicts of duties are avoided so you don't have the wolf amongst the sheep
- Security comes from people following process in their use of standards enforced techology

Authentication

- Authentication:
  - You are who you say you are
  - Involves two steps
    - An assertion of identity (account)
    - Supplemental proof that you own that identity (password)
  - Multi-factor authentication involves multiple proofs
    - Certificate
    - One time pad
    - Second password
    - Biometric
    - other

Authorization

- Someone's approval for the account to access
  - Data
  - Functionality
- Should be checked with each access attempt
  - This can change during the use of the application
  - Checking for all logic calls can prevent injection attacks
    - Is the account authorized to write to the database?
      - This is a stronger prevention than all the input filtering in the world

Confidentialities

- Many models to manage this depending upon requirements
  - Bell-LaPadula model
    - Access is granted when clearance is higher than the classification
  - Graham Denning model
    - Here there are three properties to be tracked: object, subject, and rights.
    - Rights are read, write, delete, transfer
    - Access is controlled through a lattice or matrix of property to right per account
  - Brewer-Nash model
    - Access may not be granted to objects when access is already granted to objects owned by competitors.

Integrity Models

- The Biba Integrity Model
  - Graduations of integrity,
    - where in based upon the security level of the actor, access may be granted to the simple integrity property:
    - read, or the * property where writes are permitted.
    - Writes are permitted only where the security property of the subject is equal to or higher than that of the object.
    - This prevents data of lower integrity corrupting data of higher integrity.

Integrity Models

- The Clark-Wilson integrity model is explicitly designed for the change controls suitable to a transactional system,
- no changes are permitted to unauthorized subjects
- no unauthorized changes to authorized subjects
- internal and external consistency.
- guarantees that the system performs only what is expected with every operation.
- External consistency involves enforcing that the data within the system is consistent with similar data external to the system through the application of well-formed transitions.
- Within the model, each data item is defined and changes allowed only by a limited set of operations (such as a service or application).
- These items are defined as follows:
  - Constrained data items, where data integrity is protected.
  - Unconstrained data items, where data integrity is not protected.
  - Integrity verification procedures, procedures to guarantee integrity.
  - Transformation procedures, those procedures allowed to change a constrained data item.
  - Subjects and objects are labeled accordingly
  - Procedures act as an intermediate layer between subjects and objects.
  - Through restricting access to constrained data items only through transformative procedures, data integrity is assured even when the data is changed.

Integrity Models

- Service Clark-Wilson Integrity model by Majd Mahmoud Al-kofahi as part of his dissertation entitled Service Oriented Architecture (SOA) Security Models.
- Requires a service whose function is to govern information flow, seeking out the appropriate service to perform the data transaction that he calls a root service.
- All data should be classified as either
  - constrained data items (CDI) or
  - unconstrained data items (UDI),
  - each service is assigned the tasks of processing a set of CDI.
- There are two kinds of procedures used to maintain the integrity of the service and the CDI.
  - Integrity Verification Procedures (IPV)
    - IVP are used to verify that the data and the service are in an expected state before the start of any transaction
  - Transformation Procedures (TP)
    - TP are used to perform the functions of the SOA, moving the CDI from one valid state to another.
- Transformative Procedures consist of a set of Service Transformation Procedures (STP) where each STP is the specific transformational procedures between layers of the service architecture.
- For a TP to move services from one valid state to another, all applicable STP must have completed successfully. If after all TP are performed and all IPV are still in an expected state even in an error condition, then the root service is considered to be a well formed service.

- In addition to the existing rules of the Clark-Wilson model, Al-kofahi proposes two new rule types:
- Certification Rules
  - Those rules imposed by the security officer or system owner regarding data and process integrity.
- Certain certification rules are recommended to enforce data and process integrity.
  - Certification Rule 1: All CDI are processed by a service, and each service is responsible for performing updates and modifications to its assigned CDI.
  - Certification Rule 2: Each service must have a well-defined contract that enforces all relevant certification and transaction rules.
  - Certification Rule 3: If CDI can be processed by two or more services concurrently, then all services associated with a CDI must be certified to ensure the mutual consistency of the updated CDI.
  - Certification Rule 4: Concurrent TP must be certified to maintain consistency of all services and CDI once the transformations are complete.
  - Certification Rule 5: All TP and SFT must be certified to be valid,
    - that is they must leave the CDI in a valid state, even in the case of an error.
    - To facilitate this, the security officer must make the CDI that provide input into a SFT explicit and exclusive. The identity, and its authorization to perform the transaction would be at least two of these CDI.
  - Certification rule 6: that all STPs must be certified to be part of a well formed TP
    - Must capture all the dependencies between each service and sub service called by the TP.
  - Certification Rule 8: Within each TP, the order in which STP are performed must be certified to maintain global consistency of all the services and data items.

- To permit both transaction auditing and recovery, Al-kofahi proposes certification rule 9.
- Certification rule 9: All TP and STP must be certified to write to an append only CDI (a log) all information required to both permit the nature of the transaction to be reconstructed, and if necessary facilitate recovery to a valid state in the case of an error.
- Integrity verification procedures are responsible for validating that all CDI of a SOA are in a valid state before beginning a new transaction or transformation. To ensure this, Al-kofahi proposes two more certification rules.
- Certification Rule 10: IPVs must properly ensure all CDI are valid when an IPV is run.
- Certification Rule 11: A well-formed service must be certified to ensure that all sub-service CDI remain in a valid state and that the global consistency is valid regardless of failure or error condition within any STP.
- Finally, the system must only act on CDI. If a UDI is presented to a service, it must be upgraded to a CDI or be rejected.
- Certification Rule 12: Any STP or TP that takes a UDI as an potential input must be certified to perform only valid transformations or else no transformations for any possible value of that UDI. The transformation should take the input from a UDI to a CDI or reject the UDI.

- Enforcement Rules
  - Those rules enforced by the service and transactions.
- Enforcement Rule 1: each service must authenticate the identity of all subjects attempting to execute a TP. All propagations of this identity across all dependent services for the TP must be authenticated for each STP.
- Enforcement Rule 2: Each service must maintain a list of relations that relate a subject, a TP, STP, and all CDI, which these processes may reference on behalf of each valid identity.
- Since transactions in a SOA are asynchronous, may be concurrent yet must retain the integrity of both data and process, Al-kofahi proposes a certification rule as sufficient to enforce transactional sequencing.
- To maintain the transactional sequence, and allow for both audit and recovery, each service must maintain a dependency table that records all dependencies between different services as regards to a specific TP, or use a specific service to maintain this.
- Enforcement Rule 4: Each service must maintain a dependency table recording all dependencies between different services in a service network.
- This enforcement rule is a problematic requirement in many web service applications, especially those using a Web 2.0 design.

An alternative to all these rules: Workflow

- BPEL
  - Describe the logic of business processes through composition of services
  - Compose larger processes from smaller processes
  - Handle synchronous and asynchronous operations
  - Invoke services in series or parallel
  - Selectively compensate completed activities in case of failure
  - Maintain interruptible long term transactional systems
  - Resume interrupted and/or failed activities
  - Route incoming messages to the appropriate service
  - Correlate requests within and across business processes
  - Schedule activities based upon predefined times
  - Define order of execution
  - Handle both message and time related events
- With BPEL, business processes can be described in two distinct ways:
  - As executable business processes wherein the exact details of the process are defined and follow the orchestration paradigm.
  - As abstract business processes, or the public message exchange. These are not executable, and follow the choreography paradigm.
  - In the orchestration paradigm, a central process (which can be another service) takes control and coordinates the execution of different operations in the services involved in the operation.
- WS-BPEL exists to turn this into a Workflow Service Layer

More on BPEL

- There are three basic concepts that are fundamental to successfully deploying WS-BPEL:
  - Business processes include data dependent behavior, where transactions cannot complete without key information that may be processed by another service. It is important to specify exceptional conditions where things don't go well as well as those conditions where all the data is provided as expected.
  - Long running transactions are often multiple nested processes each with their own data requirements. Business process success across a distributed system will often depend upon the coordination of the delivery of the outcome of those processes.
  - WS-BPEL is structured so that there are two basic kinds of processes,
    - Abstract
    - Executable.
      - Executable processes are fully defined and can be executed
  - This separation between abstract and executable content is similar to WSDL and the Application
  - Three areas where this is not compatible with WSDL:
    - Fault naming
    - Overloaded operations names of WSDL port types are not supported by WS-BPEL
    - WSDL port types that contain solicit-response or notification operations.

And then there is XML DSig

- Signing and Encrypting makes it secure
  - Right?
  - XMLDSig Only encrypts some of the text, only signs some of the text
  - You can replace the entire signed part of the XML
  - And no one will know – UNLESS you validate

What about the other aspects of a security architecture?

- Availability
  - Shame in a way that UDDI didn't make it, while weak from a security POV, it was very strong from a BC/DR perspective
- Utility
  - Isn't this just a well defined application?
- Control
  - Solved mostly through Authentication & Authorization
- Authenticity

WS-Security

- WS-Federation
- WS-Trust
- WS-Policy
- WS-SecureConversation
- WS-Messaging
- WS-ReliableMessaging
- WS-Coordination
- WS-Transaction

Authentication

- SAML
- Kerberos
- X509v3 Certificates
- OpenID

Privacy

- Anonymity
  - P3P
  - WS-Federation supports the ability to use a application and/or service without being tracked
  - WS-Federation permits authentication without identity
- The right to be forgotten
- Security incidents and investigations

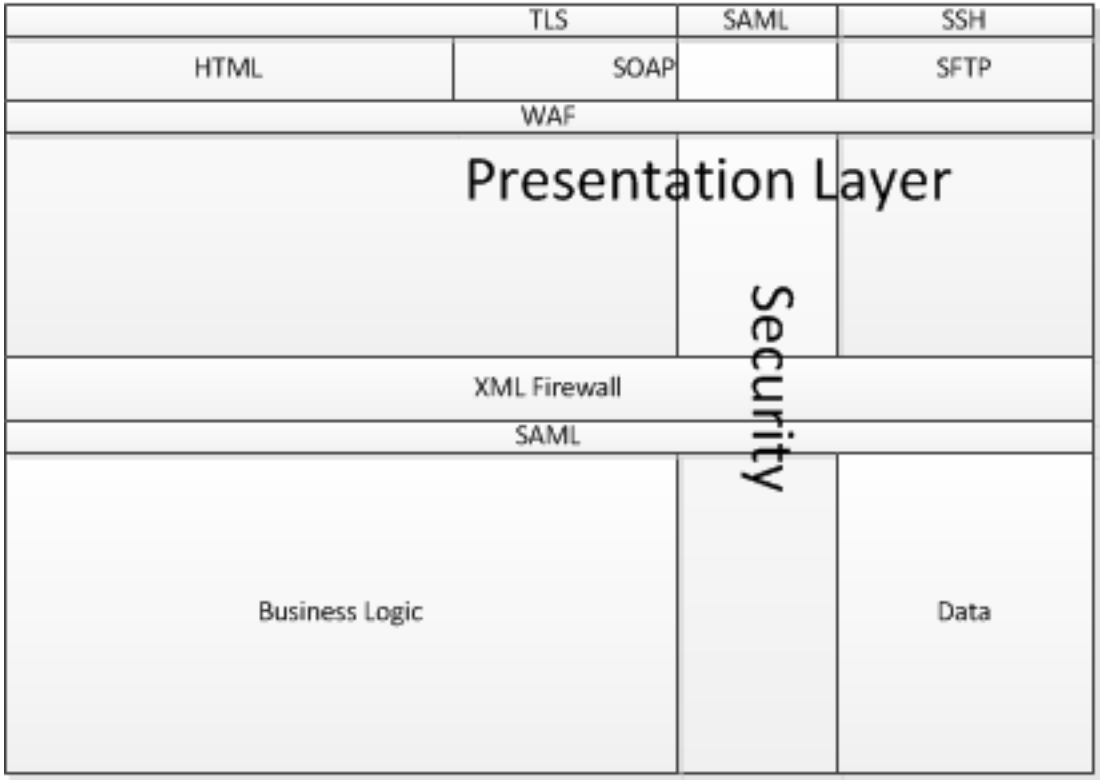
Security is another word for nothing left to loose

- Actually, Security = Quality
- Security is the assurance that the application will do:
  - Exactly what it is designed to do
  - Nothing more than it was designed to do
  - For only those people for whom it was designed to act
  - Keeping their transactions out of the prying eyes of others

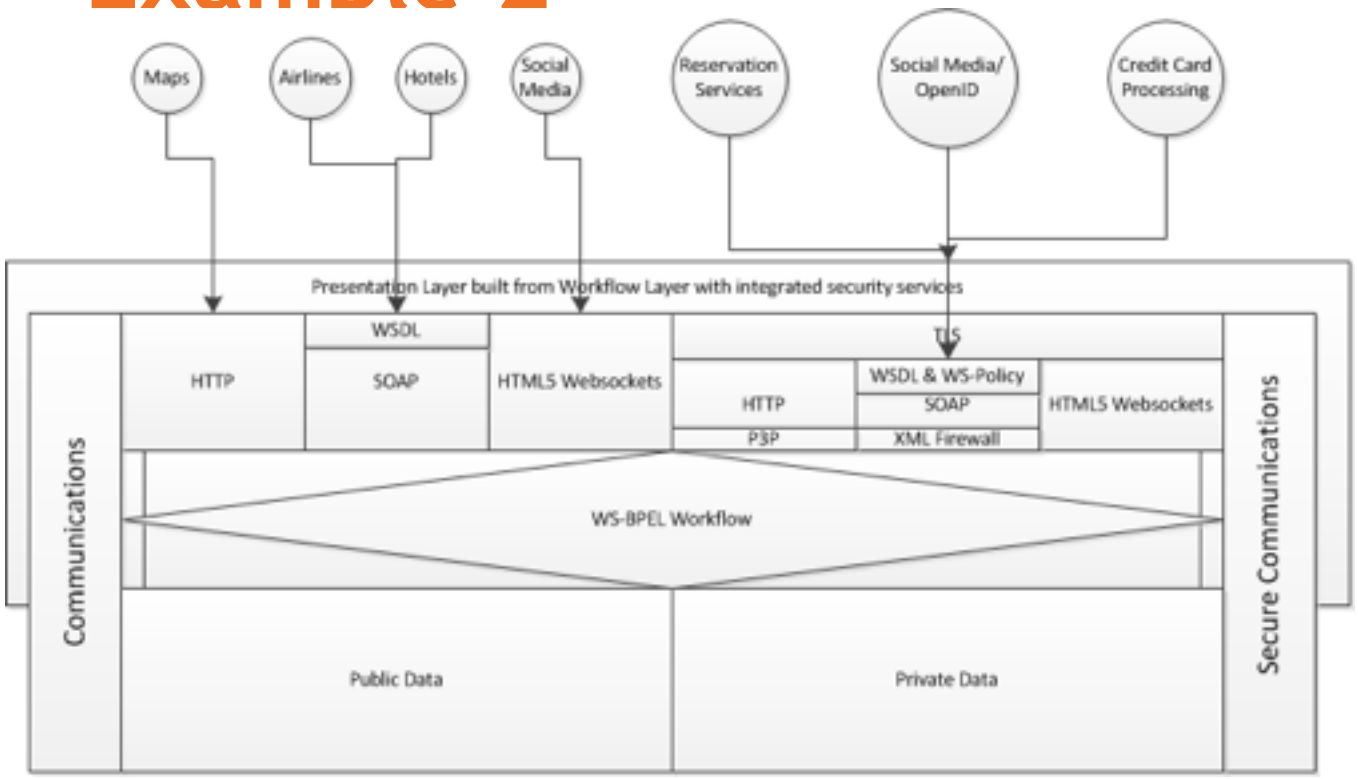
Example 1

Will be interfacing with many customers' design teams,
- These customers, who are often competitors can't access each other's designs

That the designs are not tampered with.
They provide to their customer an existing database of chips, capacitors, resistors, etcetera so that each design may be properly tested with the components that will be soldered to the board.
This means that the application has some data that is common and shared between customers and other data which is proprietary and not shared.
The customer must maintain control over who has access to their design documents and schematics



Example 2



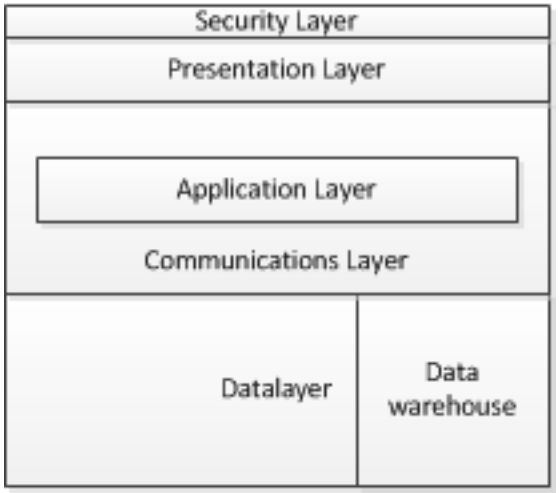
Maps will be provided by some map company, Weather information by the national weather service,
What to do by local convention and visitor's bureaus
Make hotel reservations, train and airplane reservations and book local rental car and or taxi services.
A key differentiator for this business is that it books mystery trips, where the person doing the booking has no idea where they'll be going until the transaction is complete, but you, as the service provider must be able to keep track and present enough information to allow an intelligent decision that meets the customer's budget and whimsy.
As a mystery travel service, the names and locations of the hotels, the airline destinations will be kept anonymous until after the booking is complete.

Private and personal data will be collected as part of the booking process
Because sometimes life changes and you can't keep the plans you've made, for what ever reason the customer must be able to change or cancel their reservations at any time.
You will allow various travel agencies to create accounts on your system to that they may make arrangements for their customers
The arrangements made by these agents must be able to be shared with the consumer.
Some of your customers will want to share their arrangements with family and friends via social networking sites

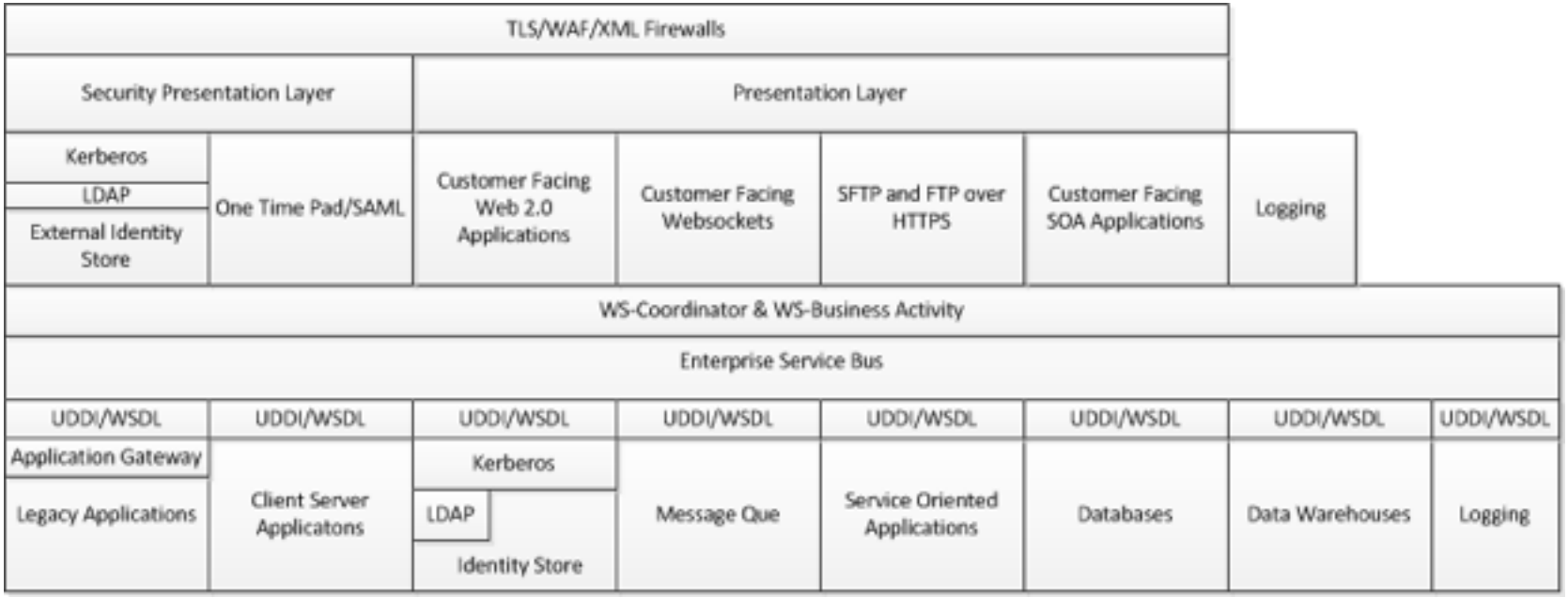
Lattice

Example 3

This is a very simple service that can be accessed by individual traders or other service providers which can wrap the information we provide into their platform.
We don't accept orders nor can our users use our service to sell.
We can, and do, however provide historical analysis for any time period since the founding of the exchange we're the front end for.
We will need to support two data feed models to support our diverse customer base: the push and the pull.
Because we don't care who accesses our service to pull data from us, there is no need for an identity service provider nor for a authentication front end.
However, we must prove that the data we represent is ours to the organizations that require us to push the prices to them.
It is for this reason that we're not supporting WebSockets at this time, as there is no mechanism to demonstrate authenticity of the data.
However, we are looking at the use of extended validated certificates to see if they meet our risk profile of our customers for sufficiently demonstrating authenticity and integrity of the data so that we can provide an HTML 5 WebSockets interface to our data.



Example 4



Extend the power of a legacy system through the Internet
The bank has deployed a message bus internally to govern and control all communications with the legacy system
A subset of the application's functionality needs to be exposed to Bank customers. This is done to individual customers through a web portal.
Corporate customers can access a business centric portal similar in design to the customer portal
Corporate customers may also access the service through service calls of their own.

Does this make your application secure

- How long is a piece of string?
  - Poodle
  - Heartbleed
  - https://yoururl.com/?wsdl
  - https://www.usenix.org/system/files/conference/usenixsecurity12/sec12-final91.pdf
- Test
  - Whitehat, Cigital, Veracode, CheckMarx, Fortify etc

For further study

- Abadi, D. Madden, S., Hachem, N. 2008 Column-Stores vs. Row-Stores: How different are they really?
- Al-kofahi, Majd Mahmoud. 2011 Service Oriented Architecture (SOA) Security Models. Iowa State University Digital Repository
- Alberts, C., Dorofee, A. 2003. Managing Information Security Risks. Addison-Wesley, Boston, MA
- Anonymous. 2007. Data Distribution Service for Real-Time Systems. Version 1.2 http://www.omg.org/spec/DDS/1.2/PDF/
- Anonymous. 2012. Workflow Services Overview. http://msdn.microsoft.com/en-us/library/dd456797.aspx
- Anonymous. 2013, Hacking Web Services with Burp. https://www.netspi.com/blog/entryid/57/hacking-web-services-with-burp
- Anonymous. 2013. MS-DCOM Distributed Component Object Model (DCOM) Remote Protocol. http://download.microsoft.com/download/9/5/E/95EF66AF-9026-4BB0-A41D-A4F81802D92C/[MS-DCOM].pdf
- Anonymous. 2006. http://uddi.xml.org/uddi-101
- Anonymous. 2013. Removal of UDDI Services from Server Operating System. http://msdn.microsoft.com/en-us/library/dd464641.aspx
- Bajaj, S., Box, D., Chappell, F. et. al. 2006. Web Services Policy 1.2 http://www.w3.org/Submission/WS-Policy/
- Bertino, Elisa. 2010. Security for Web Services and Service Oriented Architecture. Springer, New York, NY
- Boag, S., Chamberlin, D., Fernandez, M. et. al. 2010. XQuery 1.0: An XML Query Language. http://www.w3.org/TR/xquery/
- Bos, B. 1997. XML Representation of a Relational Database. http://www.w3.org/XML/RDB.html
- Bournaee, Blake. 2002. XML Security. McGraw-Hill, Berkley, CA.
- Box, D., Christensen, E., Curbera, F., 2004. Web Services Addressing (WS-Addressing) http://www.w3.org/Submission/ws-addressing/
- Bray, T. Paoli, J. Sperberg-McQueen, C.M., et. al. 2006 Extensible Markup Language (XML) 1.1 (Second Edition) http://www.w3.org/TR/2006/REC-xml11-20060816
- Brown, A., Fox, B., Hada, S., et. al. 2001. SOAP Security Extensions: Digital Signature http://www.w3.org/TR/SOAP-dsig/
- Calder, A., Watkins, S. 2006. International IT Governance. Kogan Page, Philadelphia PA
- Cantor, S., Hirsch, F., Kemp, J., et. al., 2005. Bindings for the OASIS Security Assertion Markup Language (SAML) V2.0 http://docs.oasis-open.org/security/saml/v2.0/saml-bindings-2.0-os.pdf
- Cantor, S., Hughes, J., Hodges, J., et. al., 2005. Profiles for the OASIS Security Assertion Markup Language (SAML) V2.0 http://docs.oasis-open.org/security/saml/v2.0/saml-profiles-2.0-os.pdf
- Cantor, S., Kemp, J., Philpot, R., et. al. Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0 http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf
- Cantor, S., Moreh, J., Philpott, R., et. al., 2005. Metadata for the OASIS Security Assertion Markup Language (SAML) V2.0 http://docs.oasis-open.org/security/saml/v2.0/saml-metadata-2.0-os.pdf
- Chappell, D. 2004. Enterprise Service Bus. O'Reilly Media, Sebastopol, CA.
- Christensen, E., Curbera, F, Meredith, G., Weerawarana, S. 2001. Web Services Description Language. http://www.w3.org/TR/wsdl
- Cranor, L., Dobbs, B., Egelman, S., et. al. The Platform for Privacy Preferences 1.1 http://www.w3.org/TR/P3P11/
- Deutsch, A., Fernandez, M., Florescu, D. 1998. et. al. XML-QL: A Query Language for XML. http://www.w3.org/TR/1998/NOTE-xml-ql-19980819

- Eastlake, D., Reagle, J., Solo, D. et. al. 2008. XML Signature Syntax and Processing (Second Edition) http://www.w3.org/TR/2008/REC-xmldsig-core-20080610/
- Edlich, S. Ph.D http://nosql-database.org/ Accessed September 23, 2013
- Erl, T. 2004. Service-Oriented Architecture, A Field Guide to Integrating XML and Web Services Prentice Hall, Upper Saddle River NJ
- Ferraiolo, D, Kuhn, D., Chandramouli, R. 2003. Role-Based Access Control. Artech House, Boston MA.
- Fielding, Thomas. 2000. Architectural Styles and the Design of Netowrk-based Softwre ARchitectures, Chapter 5, http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm
- Fremantle, P., Patil, S., 2009 Web Services Reliable Messaging (WS-Reliable Messaging) Version 1.2. http://docs.oasis-open.org/ws-rx/wsrm/200702/wsrm-1.2-spec-os.pdf
- Garman, Jason. 2003. Kerberos: The definitive guide. O'Reilly & Associates. Sebastopol, CA.
- Gross, Thomas. 2003. Secuirity Analysis of the SAML Single Sign-on Browser/Artifact Profile. http://www.acsac.org/2003/papers/73.pdf
- Gudgin, M., Hadley, M., Mendelsohn, N., 2007. SOAP Version 1.2 Part 1:Messaging Framework http://www.w3.org/TR/2007/REC-soap12-part1-20070427/
- Gudgin, M., Hadley, M., Mendelsohn, N., et. al. 2007. SOAP Version 1.2 Part 2:Adjuncts http://www.w3.org/TR/2007/REC-soap12-part2-20070427/
- Gudgin, M., Hadley, M., Mendelsohn, N., et. al. 2007. SOAP Version 1.2 Part 2:Adjuncts Data Model http://www.w3.org/TR/soap12-part2/#datamodel
- Herold, Rebecca. 2005. Managing an Information Security and Privacy Awareness Training Program. Auerbach Publications, Boca Raton Florida
- Hickson, Ian. 2013. The Web Socket API. http://dev.w3.org/html5/websockets/
- Hirsh, F., Datta, P. 2012. XML Signature Best Practices http://www.w3.org/TR/2012/NOTE-xmldsig-bestpractices-20120710/#denial-of-service
- Hirsh, F., Philpott, R., Maier, E., 2005. Security and Privacy Considerations for the OASIS Security Assertion Markup Language (SAML) V2.0. http://docs.oasis-open.org/security/saml/v2.0/saml-sec-consider-2.0-os.pdf
- Holmes, Troy. What is application Architecture. 2013 Conjecture Corporation. http://www.wisegeek.org/what-is-application-architecture.htm
- http://db.csail.mit.edu/projects/cstore/abadi-sigmod08.pdf
- http://lib.dr.iastate.edu/cgi/viewcontent.cgi?article=3006&context=etd
- http://msdn.microsoft.com/en-us/library/bb676633.aspx Accessed on September 1, 2013
- http://msdn.microsoft.com/en-us/library/f8e50t0f(v=vs.85).aspx Accessed on September 1, 2013
- http://msdn.microsoft.com/en-us/library/live/hh826544.aspx Accessed on September 1, 2013
- http://sourceforge.net/p/ws-attacker/wiki/Home/ accessed on 9/1/2013
- Smorovsky, J., Mayer, A., Schwenk, J., et. al. 2012 On Breaking SAML: Be Whoever you want to be. https://www.usenix.org/system/files/conference/usenixsecurity12/sec12-final91.pdf

- Imamura, T., Dillaway, B., Simon, E. 2002 XML Encryption Syntax and Processing. http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/
- INCITS/ISO/IEC 27001-2005 2006 Information Technology – Security techniques – Information Security Management Systems – Requirements. Information Technology Industry Council (ITI)
- Johnson, Ken. 2011. Attacking Web Services Pt 1 - SOAP. http://resources.infosecinstitute.com/soap-attack-1/
- Johnson, Ken. 2011. Attacking Web Services Pt 2 - SOAP. http://resources.infosecinstitute.com/soap-attack-2/
- Jordan, D., Evdemon, J., 2007. Web Services Business Process Execution Language Version 2.0 http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html
- Juric, M. 2006. Business Process Execution Language for Web Services. Packt Publishing, Burmingham UK
- Kaeo, Merike. 1999. Designing Network Security. Macmillan Technical Publishing, Indianapolis, IN
- Kaeo, Merike. 1999. Designing Network Security. Macmillan Technical Publishing, Indianapolis, IN
- Kakimus, N., Bradley, P., Jones, M., et. al. 2013. OpenID Connect Basic Client Profile 1.0 - Draft 28 http://openid.net/specs/openid-connect-basic-1_0-28.html
- Kaler, C., McIntosh, M., 2009. Web Services Federation Language (WS-Federation) 1.2 http://docs.oasis-open.org/wsfed/federation/v1.2/os/ws-federation-1.2-spec-os.html
- Kemp, J., Cantor, S., Mishra, P. et. al., 2005. Authentication Context for the OASIS Security Assertion Markup Language (SAML) V2.0 http://docs.oasis-open.org/security/saml/v2.0/saml-authn-context-2.0-os.pdf
- King, C., Dalton, C., Osmanoglu, T. 2001 Security Architecture. McGraw-Hill, Berkley, CA.
- King, C., Dalton, C., Osmanoglu, T. 2001 Security Architecture. McGraw-Hill, Berkley, CA.
- Krafzig, D., Banke, K., Slama,D. 2005. Enterprise SOA, Service-Oriented Architecture Best Practices. Prentice Hall Upper Saddle River NJ
- Kurtzban, Stanley. 2009 "Inplementation of Access Controls" 432-447 In Information Security Management Handbook. CRC Press LLC, Boca Raton Florida
- Landoll, Douglas. 2006. The Security Risk Assessment Handbook. Auerbach Publications, Boca Raton Florida
- Lawrence, K., Kaler, C. 2004 Web Services Security. https://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf
- Lawrence, K., Kaler, C., 2007. WS-SecureConversation 1.3 http://docs.oasis-open.org/ws-sx/ws-secureconversation/v1.3/ws-secureconversation.html
- Mandia, K., Prosise, C., Pepe, M. 2003. Incident Response. McGraw-Hill, NY.
- Meier, J.D., Hill, D., Homer, A., et. al. 2009. Microsoft Application Architecure Guide, 2nd Edition. Microsoft Press. http://www.microsoft.com/downloads/details.aspx?FamilyID=ce40e4e1-9838-4c89-a197-a373b2a60df2&DisplayLang=en
- Nash, A., Duane, W., Joseph, Brink, D. 2001. PKI: Implementing and Managing E-Security. McGraw-Hill, NY.
- Newcomer, I., Robinson, I., 2009. Web Services Atomic Transaction (WS-AtomicTransaction) Version 1.2 http://docs.oasis-open.org/ws-tx/wstx-wsat-1.2-spec-os.pdf
- Newcomer, I., Robinson, I., 2009. Web Services Coordination (WS-Coordination) Version 1.2 http://docs.oasis-open.org/ws-tx/wstx-wscoor-1.2-spec-os.pdf
- O'Reilly, Tim. 2005 What is Web 2.0. http://oreilly.com/web2/archive/what-is-web-20.html
- Parker, Donn. 1998. Fighting Computer Crime, John Wiley & Sons, New York, N.Y.
- Peltier, Thomas. 2002. Information Security Policies, Procedures, and Standards. Auerbach Publications, Boca Raton Florida
- Preibush, S., 2006., Privacy Negotiations with P3P. http://www.w3.org/2006/07/privacy-ws/papers/24-preibusch-negotiation-p3p/
- Reese, George. 2009 Cloud Application Architectures. Sebastapol, CA
- Rosenberg, J., Remy, D., 2004. Securing Web Services with WS-Security. SAMS Publishing, Indianapolis IN
- SAP News Desk. 2005. Microsoft, IBM, SAP to Discontinue UDDI Web Services Registry Effort. http://soa.sys-con.com/node/164624
- Shreeraj, S.. 2007. Hacking Web Services. Charles River Media, Boston MA