APPLICATION AUDITS: **GET THE BIG PICTURE**

Proper risk assessment reveals what to—and what not to—audit

By Scott Collins, CIA, CISA, MCIS, MCOM, MBA

As auditors, we have to assess the risks before beginning an audit. We have to know enough to assess the *relevant* risks. And you won't know enough until you make some assessment of the big picture. The assessment will tell you not only what not to audit, but also how risky the risks are.

One way to know the big picture when auditing an application is to look at its architecture—specifically, its server-network architecture. An application is usually a set of server and network devices working together to provide application functionality to the end-users. The application architecture will tell you some very important details, such as what parts of the application:

- Are accessible to users
- May be more critical than other parts
- May have important dependencies

Let's look at why these items matter from an architecture perspective, for IT auditors and for operational and financial auditors as well.

CONTROL AREAS

The three primary areas of control when examining an application and its architecture are input, processing and output.

User access – The first area relates directly to input controls. If there is no access, there cannot be any input. User accessibility can also relate to output controls.

Criticality – This second area relates to processing controls. An application architecture diagram can show which parts of the application provide processing that is most important.

Important dependencies – This final area relates most to availability. Availability is a key area of control that can affect all other application controls, including the primary controls of input, processing and output.

Proper risk assessment and scoping demands that you know what all the relevant risks are, how severe those risks are, and particularly what not to audit.

Before looking at each of these big picture areas from an architecture perspective, let's first review some basic architecture concepts.

An application's architecture can be quite complex. The example diagram is a relatively simple one, but enterprise applications, those which serve an entire company, can consist of many network and server components.

Often segmented by tiers, as shown in the diagram, each tier performs a different function:

- The presentation tier presents how the application will look to the end-user.
- The middle tier handles the main business logic or processing of the application.
- The data tier directly manages the application data.

Most of today's applications use a client-server model. The client workstation runs part of the application, often a small part, and servers run the rest of the application. Some of these servers may support more than one application. A good example is the authentication server shown in the diagram. As explained later, authentication servers are often used to provide user authentication services to multiple applications.

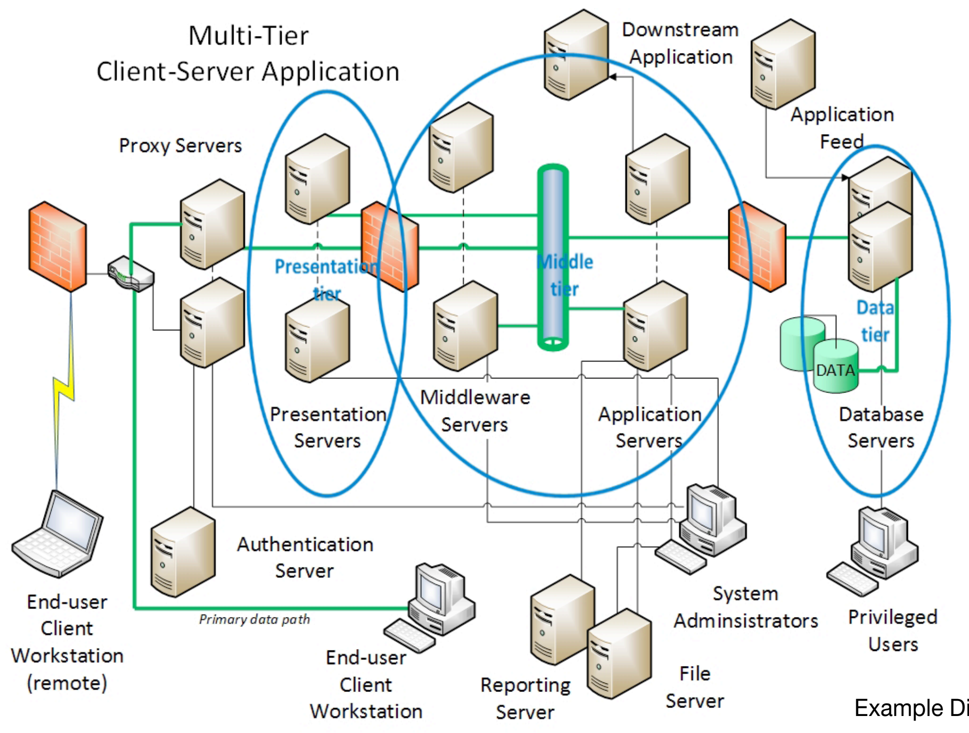
With this basic understanding of the multi-tier, client-server application model, let's review the parts of an application from an architecture perspective.

USER-ACCESSIBLE PARTS

Based on the application's architecture, every network and server component that you see in an architecture diagram is a potential user access point. The user can be a person or non-person. The non-person user is often an external application using an ID and password to authenticate and access that part of the application. The non-person user may be internal to the application, such as when one server needs to authenticate and access another server, all within the same application.

Generally, access points internal to the application, (i.e., server to server) do not pose much security risk. Normally only IT support personnel, such as System Administrators, have access to the internal servers and network components of an application.

Other access points may pose great risk. As shown in the architecture diagram, most end-user clients would access the application from the front end, the Presentation tier (left side of the diagram).



Example Diagram

Other "privileged" users may access the application at the Data tier (right side of the diagram). Such privileged access allows direct access to application data without logging into the application's presentation interface. The presentation interface is usually a graphical user interface (GUI), often presented as a web page by a web browser or other client application. Privileged user direct access to application data bypasses application access controls, although other access controls may be applied by the Data tier servers.

Access points can also include points of output. These are where users can access and receive output from the application. Sometimes these "users" are really file, print, reporting or other servers that receive output on behalf of application end-users. Sometimes the output user can be a downstream application that does further processing of the data. (See bottom and middle-tier areas of the example diagram.)

Some output access points may not be very important in an application audit. To illustrate, a file server, which may technically be part of the application, may not be as important as the direct screen output sent by the application to the end-user client workstation. The file server may just provide file access in addition to this screen output. The access points to ultimately address should be part of an audit's initial risk assessment and scope decisions.

PARTS MORE CRITICAL THAN OTHERS

As mentioned earlier, some application parts relate to processing controls. An application architecture diagram can show parts of the application providing processing controls that are important.

In a system network architecture like the one in the example diagram, the application architecture may only suggest major areas of processing. It cannot show processing or functions that are internal to the application, but a dataflow or workflow diagram may show this.

Knowing where the major areas of processing lie can help you understand how distributed or concentrated the processing is, what specific servers are involved, and perhaps the type of processing. In the example diagram, which is typical of many basic multi-tier, client-server applications, most processing is handled by the middle tier, specifically middleware and application servers.

By definition, middleware servers apply business logic or "rules" to user requests and help determine how requests should be processed by the application servers. Middleware servers may also service other applications' servers. The application servers are normally the primary host or execution point of application functionality.

Apart from the middleware and application servers, some processing may be done on the Presentation tier servers in order to properly present the application to the end user, depending on the computing device they use (tablet, mobile phone, or laptop/desktop computer). In addition, some processing may be performed by the Data tier servers. These are often database servers using stored procedures, meaning stored programs that process and manage the application data.

Finally, depending on the application, peripheral or co-processing may be performed by other servers within or outside the application's architecture. The extent of such processing may vary greatly depending on the application's overall functional design.

Much of the application processing cannot be seen from the architecture diagram, so processing controls may need to be assessed by looking at the application code, or doing input-output testing. As with access points, what processing to consider should be part of your application audit risk assessment and scope decisions.

IMPORTANT DEPENDENCIES

In addition to examining an application's architecture for input and output and processing, an architecture diagram illustrates important dependencies that directly affect application availability. Such dependencies are primarily illustrated by connections between the servers of the application. Most important are the connections that are on the primary data path between the end-user and the data repository (shown in the diagram as a green line).

In general, this path is from the front-end Presentation tier servers, to the middle tier servers, then to the Data tier servers,

The architecture can show important dependencies.

then back to the end-user client workstation. However, the primary data path may not be obvious, particularly if many servers are involved. You may want to discuss this data path with the application support team. When the primary path is identified, if any connections along this path are broken, important parts of the application, if not the entire application, will be unavailable to the end-users.

You also will want to note those servers that are not on this primary data path. These servers may still be critical to the application if they serve an important function. For example, as seen in the diagram, an authentication server is often used to provide user authentication to the application. If this server(s) is unavailable, no one will be able to log in to the application and probably other applications as well.

In addition, notice in the example diagram the use of an application feed to the database servers (top right of diagram). While not on the primary data path between the end users and Data tier servers, if this feed is lost, some application data may not be available.

One more example is the reporting server (bottom middle of diagram), which is used to provide application reports to end users. You should ask how important is this server and the connection to it. Loss of this server, or connection, may deny end-user access to critical reports. Some other dependencies may require further discussion with the application support team.

Dependencies are best shown by the connections between servers, but the servers themselves can also represent dependencies affecting end users. An architecture diagram will often show redundant servers.

Since application and middleware servers are usually critical to an application's processing, multiple servers may be needed to ensure continuous operation. However, there may also be stand-alone or single servers. Such servers, as in the case of a single authentication server, can represent a potential single point of failure that can cause an entire application to become unavailable. You may need to discuss server redundancy with the IT application support team, but at least knowing there is, or is not, server redundancy may help you decide whether further investigation is needed.

The bottom line is that important dependencies may affect critical application functionality involving input, processing, and output. Reviewing the application's architecture can help the auditor assess this risk and scope the audit appropriately.

CONCLUSION

When performing an audit, the big picture is important. An application's architecture is one of the best ways to get that big picture. Proper risk assessment and scoping demands that you know what all the relevant risks are, how severe those risks are, and particularly what not to audit.

The relevant, identified risks are usually associated with the application's primary input, processing, and output controls. An application's architecture can illustrate these risks and controls by showing access points and where the processing is performed, mainly by server type and tier location.

Most of all, the architecture can show important dependencies, especially along the primary data path. Server connections and server redundancy are important factors to consider when examining dependencies. For certain critical servers that lie off the primary data path, the application may be just as dependent on these servers and their connections. Ultimately, important dependencies can affect the application's overall availability.

Before doing an application audit, you should take a quick look at its architecture—its access points, processing servers, and server-connection dependencies. This perspective will be time well spent before diving into testing the specific input, processing and output controls. DI

Scott Collins is a Senior IT Auditor in Enterprise Risk Management Services at Providence Health & Services located in Renton, Wash. He has been an IT Internal Auditor for nine years, and has reviewed application architectures for their compliance to corporate and infrastructure best-practice security standards. You can reach Scott at SCollins.ahia@nym.hush.com.