Five Key Questions Developers Need to Ask before Starting the Troubleshooting Process

Lori MacVittie, 2009-06-05

Brother, can you give a developer a hand?

As the topology of networks delivering applications becomes increasingly complex it becomes more and more difficult to troubleshoot problems, especially for developers tasked with figuring out why their “application broke” in production when it was working just fine thank you very much in “DEV” and “QA.” It is rare, after all, that the production environment – including all the moving parts – is duplicated in development and testing environments.

It is already difficult enough for developers to track down problems due to the complex nature of application infrastructure stacks. It is a rare developer that isn’t already abstracted several layers from the actual network, and rarer still one that isn’t dealing with at least three tiers of an architecture. It’s a given that data flows to and from the client through several devices, but understanding how many of them may alter/modify/change the data is imperative to helping developers track down issues and resolve them as quickly as possible.

THE MEN IN THE MIDDLE

Multiple components within a highly scalable architecture modify – if not data – then the HTTP headers through which quite a bit of functionality is delivered within the data center infrastructure. Load balancers may add or remove cookies, insert additional headers or replace existing ones with their own. Caches may add tags or redirect requests to content delivery networks (CDN). Content may be transported partially via SSL and then in the clear. Authentication headers may be added and removed by a variety of devices.

And they’re all outside the control of the developer. Worse, they may even be outside the awareness of the developer in that the developer may not be aware of them at all. Many of these functions are considered the realm of network specialists. While from an operational point of view this may very well make sense but from a developer’s point of view anything that may affect the flow of data between client and server needs to be detailed, documented, and available for reference.

The order in which network-hosted application-focused services are applied to data can change not only the behavior of the application (usually to something unexpected and undesirable) but the behavior of other infrastructure devices as well. When it is the task of any device to inspect application layer data and compare it against known states – whether for security or performance or related functionality – then changes to the order of that flow can be paramount to putting the data in a blender and asking the device to put it back together.

The opposite is true, as well. For example, if you are using custom HTTP headers to transport some application information and the architecture has implemented a strict protocol security component, that component may flag your application data as being “malicious.” It’s important to understand what components exist that act on or inspect and apply policies to application data in order to ensure proper configuration of all components and a well-behaved application and infrastructure.

In today’s world of increasingly complex architectures the behavior that occurs because of interactions between application delivery and infrastructure systems when an application is deployed into a production environment can be nearly impossible for a developer to replicate. This makes it nearly impossible to troubleshoot, which means the problem may not be resolved for months – if ever.

WHO. WHAT. WHERE. WHEN. and WHY

WHO, WHAT, WHERE, WHEN, and WHY

Developers – and network professionals too, for that matter - need to know five key things in order to streamline the troubleshooting process.

1. **Who**
   Who is responsible for managing each of the devices in the flow of data. This information will be useful if the developer needs to look at a log or needs additional, more detailed, logging at any given point in the flow of data. Spending hours tracking down someone who can help obtain this information is time wasted while customers – and potentially revenue – is lost. This is increasingly important as you really can't "step over/into" the network (yet). Troubleshooting in a complex networked environment almost necessarily requires that you fall back on ancient techniques of logging and "print/echo/>>" in order to track down issues.
2. **What**
   What is the flow of data through the system? This needs to include every device that may modify the data, even if it only does so based on certain conditions. What criteria is used should also be understood by the developer, as conditional network-based policies that are rarely invoked can certainly cause intermittent application problems.
3. **Where**
   Where are modifications made? For example: which device, if any, is responsible for inserting the X-Forwarded-For and can it/does it maintain the original client IP address elsewhere in the headers?
4. **When**
   When is the modification made? Is it made on ingress? Egress? Are the policies invoked only at certain times of the day? The week? Based on certain thresholds of traffic/CPU utilization?
5. **Why**
   Why is a modification to the data made? Understanding why a modification is necessary may lead to developers offering up an alternative – perhaps even less impactful on the network – solution. For example, the modification of caching tags may not be necessary if developers or web administrators make careful use of caching and other content-freshness specific meta-data on the web or application server.

In the event that a problem occurs with an application, it may very well be the case that one of the application or application network components is responsible because it modified the data. You and I both know – probably painfully and from experience – that even when this is the case the person who has to figure it out is the developer. Because after all the *application* wasn't working as expected, regardless of whether the problem is with the actual code or because some other component changed the data mid-stream. And who is responsible for the application? Yup. The developer.

Understanding what – if any - modifications that may be made by the application and application network infrastructure may short-circuit the troubleshooting process if the problem is obviously one related to some specific piece of data or protocol header that is changed by the infrastructure. And remember the network and operational guys are people, too, and misconfiguration and conflicting rules in policies and devices happens – just like defects/bugs happen in code. But developers need to proactively seek out this information and keep it up to date in order to ensure that you have an accurate picture of what the network is doing to application data.

Asking these simple questions – and getting the answers – may be the key to shortening the time it takes to troubleshoot a tricky application problem from weeks to hours.

THINK THIS IS HARD? WAIT UNTIL YOU GET TO THE CLOUD

It can only get worse. I know asking the network guys for all this information isn't as easy as it sounds, but at least you *can* ask, even though it may be painful and may involve bribes or live chickens. Or both. Consider how much more difficult it will be when (or if) your application is running in the cloud, on an "abstracted" infrastructure that "you don't need to know or care about."

Yeah. That was *my* thought, too.

The claim that the end-users don't need to understand the infrastructure "in the cloud" is bunk and just plain wrong. If you're developing applications and deploying them in a networked environment (that means everyone, doesn't it?) then you *need to know* and understand that network.

Anything else is just asking for long, thankless nights of debugging application problems that may very well end up to be "network" problems anyway.