The Fix Must Occur by Rewriting the Code. Wait, What?

Lori MacVittie, 2007-06-11

This article is just full of interesting ideas.

First we're told that the **only** way to secure Web 2.0/SOA/Web applications is to ***rewrite the code.*** This "rewrite the application code" to address any number of delivery issues - security, performance, availability - is old and busted. There are other more efficient mechanisms that can certainly be used to address application delivery issues, such as an *application delivery network* comprising appropriate intelligent, application aware devices capable of ensuring that all applications are fast, secure, and available. These solutions *do not* require that the application be rewritten, and in fact in many instances rewriting the application **will not** solve the problem because some of the issues related to availability, security, and performance are the direct result of protocol inefficiencies and vulnerabilities (HTTP, TCP, IP) that cannot be addressed by rewriting application code. That's because the network and application stacks are **not under the control of the application developers.**

Michael Sutton, security evangelist with SPI Dynamics, now part of Hewlett-Packard Co. (HP) speaks out on Web application security.

> He said companies have always operated under the assumption that IT is responsible for security and not the Web developers. The problem is that once faulty applications are launched, IT can't provide the fix. The fix must occur by rewriting the code. But there are ways IT can help the developers get it right.

Are you ***really*** suggesting that application developers rewrite the Java TCP/IP stack to address inefficiencies and vulnerabilities? Are you ***really*** saying that the only way to deal with language-specific vulnerabilities (ASP, PHP, JSP, etc...) is for the application developers to rewrite the interpreters executing the application?? Are you really implying that there is **no fix** IT can provide other than flogging of developers?

Come on, this is one of the reasons Web Application Firewalls exist - to address those vulnerabilities that simply can't be addressed by the application developer whether due to location in the application/network stack or the time and expense of rewriting the application.

The article gets stranger (which I wasn't sure was possible) when Josef Brunner, security solutions manager at Enterasys Networks, starts discussing SOAP-based security issues.

> Brunner expressed particular concern for how the Simple Object Access Protocol (SOAP) is used in Web services. SOAP is a way for a program running in one kind of operating system such as Windows 2000 to communicate with a program in the same or another kind of an operating system such as Linux by using the Hypertext Transfer Protocol (HTTP)and its Extensible Markup Language (XML) as the mechanisms for information exchange.
>
> *A rather simplistic definition but not completely off-target. We call it "loose coupling", and its the cornerstone of what makes SOA work.*
>
> SOAP is platform-independent and allows users to bypass whatever security devices are on the network, Brunner said, adding that encryption tends to be the only security mechanism for SOAP. "SOAP is very flexible and dynamic, which is always bad from a security standpoint," he said.

> *Wait, what? SOAP does not "allow users to bypass" security devices on the network. If users/clients are bypassing security devices on the network in a SOA (Service Oriented Architecture) then the enterprise architects have failed at designing and implementing a secure, robust SOA. SOAP doesn't "allow" such a thing any more than any other application "allows" such a bypass to occur.*
>
> *And if "encryption tends to be the only security mechanism for SOAP" then implementors aren't paying attention to the myriad web services standards available from OASIS that provide for authentication and authorization specifically for Web Services (WS-Security 1.1), as well as message and field level encryption (XML Encryption) and non-repudiation (XML Digital Signatures).*
>
> *If only there were XML/SOA security solutions that could more efficiently screen traffic by acting as a reverse proxy (endpoint) and enforcing organizational security policies regarding authorization, authentication, and message contents. If only!*
>
> SOAP tends to be encrypted by an inconsistent set of methods and so there's no way for security professionals to break and inspect the traffic for trouble. Making matters worse, he noted that SOAP servers are connected to critical back-end systems attackers can compromise with the right exploits.
>
> *SOAP messages tend to be encrypted using industry standard encryption a la SSL.*
>
> Brunner's suggestions for improving the situation include securing SOAP servers with host-based IDS to prevent buffer overflow attacks, and, above all, demanding better application security, which means training developers to do better.
>
> *Wait, what? This is a complete non-sequitor. Let's burden SOAP servers with even* more *resource intensive processing (IDS) that require additional maintenance and costs to deploy and manage. It's not like processing XML is CPU and memory intensive or anything. It's not like AJAX-based applications aren't sucking up a ton of overhead and entries in the session state table because of long-lived sessions and additional connections.*
>
> *An IDS is not going to solve authentication/authorization issues, it's not going to solve the encryption problem, and they are not necessarily going to be able to deal with application-specific vulnerabilities. Besides, it's really difficult to convince a developer that you should be deploying agents on servers that will likely significantly degrade the performance of their application.*

If ***only*** there were ***some*** sort of network devices that could deployed ***in front of*** SOAP servers that not only optimized and accelerated protocols like HTTP and TCP but could also prevent buffer overflow attacks and application-specific vulnerabilities by acting as the first line of defense at the network perimeter. If only there were solutions to these problems that didn't involve rewriting applications (time, resources, money) or deploying solutions that don't address ***all*** the issues (IDS).

It's true that in general developers need to be more security conscious. It's also true that there are specific types of vulnerabilities that ***cannot*** be addressed outside of the application at this time (application flow/logic errors are peculiar to the application). But it's patently ***untrue*** that IT "can't fix the problem" because there exist both Web Application Firewalls as well as holistic application delivery networks that provide excellent solutions that address both security ***and*** performance issues associated with SOA/XML-based applications. Ensuring SOAP services within a SOA are deployed within a robust, dynamic application and network architecture **is the task of an integrated and cross-functional team from IT**, not the individual developer of services.

There is often more than one answer to the problem of application security, and though "rewrite the app" is ***always*** one of those options, it's rarely the most efficient or cost-effective option out there.

Imbibing: Pink Lemonade

Technorati tags: F5, MacVittie, security, SOA, application delivery, web applications