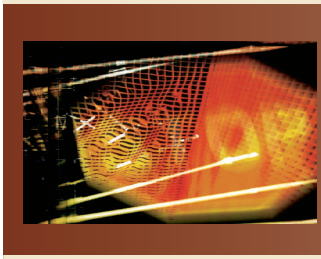
The Untrustworthy Web Services Revolution

Ken Birman, Cornell University



Without secure and robust data replication, Web-based systems may be heading for disaster.

The commoditization of the Web is bringing tremendous benefits, but also some serious risks. The benefits are obvious: Web-based technologies are becoming a universal standard. At the same time, however, the economics of the Web favor an untrustworthy technology base.

In many critical settings, such as computer networks used for medical and financial data management or for controling the electric power grid, a wave of insecure, unreliable, and otherwise deficient solutions will soon be deployed. Nonetheless, these solutions embody best-of-breed technologies in accordance with best common practice.

The key driver for this concern is service-oriented architectures. Bill Gates has suggested that a services technology revolution is under way that will ultimately dwarf the original Internet boom. Major database products, embedded systems platforms, and turnkey solutions in areas ranging from process planning to supply-chain and customer-relations management are adopting a service-oriented approach.

Web services provide the most visible example of Gates' vision—and its underlying dangers.

APPEALINGLY SIMPLE

The basic idea of Web services is simple. Browsers use a client-server style of computing in which document retrieval plays the role of method invocation, arguments are encoded into the URL, session identifiers are kept in cookies, and results are in the document that the server sends back. Web services replace the browser with a program and layer remote procedure calls (RPCs) over these XML- and HTTP-based standards.

Of course, there's more to it than that. For one thing, encoding RPC requests into SOAP (the Web services XML standard for service invocation), making a TCP connection to the remote server, and then sending requests using HTTP is inefficient.

The Web services architecture also addresses all sorts of things that don't arise with the use of browsers: transactions, request queuing, publish-subscribe event notification, standards for documenting APIs and encoding the associated type information into Web Services Description Language stubs, describing a service using the Universal Description Discovery and Integration standard, and so on.

Yet despite all the bells and whistles, the correspondence between browser and Web site remains very visible. Indeed, you can literally point a normal Web browser at a Web service and interact with it through a Web page, sending requests by filling in a form and submitting it.

Moreover, developing Web services (and client systems) is incredibly easy: With any standard development tool, including the most popular Windows and Java development platforms, the user can create a new Web service or client system at the touch of a button.

A HOLY GRAIL?

The services computing model is already a runaway success, yet the revolution has barely begun. Web-services-based computing systems are finding their way into increasingly sensitive roles: medical systems that manage patient care records and link providers with hospitals and pharmacies, computer-assisted air-traffic control systems, systems that administer energy delivery and electric power grids, chemical refineries, and rail systems.

Using a standard called Net-Centric Enterprise Services, the Global Information Grid (http://ges.dod.mil), a massive networking and integration project, is about to deploy Web services as a universal standard throughout the US government and military.

Today we're surrounded by special-purpose computers with limited functionality, such as cell phones, and a diversity of nonstandard point-to-point communication options. Tomorrow, however, everything will be on the Internet. Web services will cover the stack from the smallest sensors to massive data centers run by companies like Amazon.com and Google—indeed, both have already adopted Web services specifications.

By enabling almost anything to talk to just about anything else, Web services may be a kind of Holy Grail for the industry, tearing down what has been a pervasive barrier and ushering in a huge new wave of integrated solutions.

UNDER ASSAULT

But this leads to a basic problem. We need computers that can be *trusted*. This term is more appropriate than "secured" because the Web services revolution comes on the heels of a profound failure in the area of security.

For all the hype about more secure versions of the major platforms and popular products, and the heavy investment in safeguarding the Internet, security has been a catastrophe.

Right now, security means that I can do a Web transaction without anyone reading my credit card number out of the messages, and that if I employ an arcane assemblage of virus scanners, firewalls, spyware removal tools, and spam filters, my machine won't get infected very often.

Yet we're under a barrage of innovative assaults. Phishing, spoofing, and spam are only the tip of the iceberg. Hackers are downloading credit card and Social Security numbers from all sorts of "secure" databases. Popular sites are under continuous distributed denial-of-service attacks.

We're also seeing wave upon wave of viruses, to the point that many users have become almost indifferent to the issue. The most virulent virus to date infected several million machines in about 20 minutes, but hundreds of millions of machines could someday be compromised in seconds if the Internet's vulnerabilities remain unaddressed.

Moreover, the most virulent viruses have been fairly benign. A virus that could physically damage massive numbers of computers is entirely feasible. How long will it be before a terrorist—or even a high school student—designs a virus that could destroy hundreds of billions of dollars worth of hardware in a few seconds?

An entire black market has sprung up around tools for breaking into Web sites and end-user systems. Spyware was unknown a few years ago; today it's ubiquitous. Desktop machines are infested with the stuff, and keystroke loggers are legion. It isn't even hidden: Venture capitalists are funding companies to develop tools that capture and scrutinize consumers' actions to obtain new marketing insights.

FLIRTING WITH DISASTER

It doesn't take an oracle to see that the mania for Web services, combined with such rampant online threats, contains the seeds of a future debacle. We're poised to put air-traffic control, banking, military command-and-control, electronic medical records, and other vital systems into the hands of a profoundly insecure, untrustworthy platform cobbled together from complex legacy software components.

Vendors are aggressively rolling out a technology that could make it easy for almost anyone to tap into just about anything.

If spyware slows down my PC, that's inconvenient. It's a far more serious matter if vulnerabilities allow an intruder to wire-transfer my retirement savings to Nigeria, kill a patient in an intensive care unit, or launch a cruise missile from a Navy warship.

Vendors aren't blind to the issue: Bill Gates has singled out security and better self-management tools as the most pressing priority for Microsoft. Yet in the tension between new product and security features, hot new ideas usually win out. "Market failure" is often cited as the main reason that modern computing platforms, including Web services, offer such desultory options for security, reliability, scalability, guaranteed responsiveness, and self-administration.

A MULTIFACETED PROBLEM

Breaking the cycle is going to require a response on many levels. The problems we're confronting have ethical, legal, and economic dimensions as well as technical ones:

- Why do kids view breaking into computer systems as a game?
- Why aren't we insisting that operators of sensitive computing systems have an obligation to maintain security, and forcing them to carry liability insurance to compensate anyone damaged by their failure to do so?
- Why is the technology economy so focused on software product quality on a per-product basis and indifferent to the inadequacies of systems built by integrating components using those products?

A significant issue stems from the security community's own narrowness. For almost two decades, security has been hijacked by a group of researchers who see the field as centered on cryptography and information flow, but utterly unrelated to other kinds of trusted-computing issues such as guarantees of availability, quality of service, or correctness.

Moreover, closely related technologies play key roles in the flagship Web services products from two of the largest platform vendors. Data replication is pervasive, yet end users have absolutely no way to access these kinds of tools in traditional platforms.

It may seem odd that vendors would recognize the need for replication technologies for their own use, yet deny consumers and application developers access to those same solutions. The reason, ultimately, is that the market isn't willing to pay a premium for these kinds of solutions, hence vendors seeking to sell products in high volumes at low cost aren't offering them.

But what makes replication such a big deal? A tremendous number of trustworthy computing problems come down to replicating forms of information in a secure and *robust* manner. Traditional security addresses only half the issue. The reason that developers need replication mechanisms is that only replication can ensure access to critical data in the event of a fault, and offer a means to build components that react in a coordinated way after disruptions.

ANATOMY OF A FAILURE

To see how this dynamic can play out, consider the electric power grid. Two decades of restructuring have given us a competitive power marketplace shared by producers, consumers, independent service operators that control long-distance lines, companies that deliver power to our homes, and even micropower producers.

In this context, "security" refers to the quality of the electric power delivered to consumers: The power grid is secure if it has the appropriate voltage and line-frequency characteristics. The August 2003 power failure on the US east coast, as well as the fall 2003 European blackout, illustrated the sense in which insecurity has crept into the restructured power grid.

Both crashes were ultimately associated with software inadequacies. For example, the US blackout had a fairly routine origin: A tree fell on a high-tension cable. But the regional monitoring software had crashed, and the status displays weren't showing updates. The total lack of a global-scale monitoring infrastructure exacerbated the problem—neighboring operators had no way to make sense of what they were observing except by telephone.

This structural deficiency reflects the broader absence of software tools for monitoring large-scale systems of all kinds, including Web services, with the mixture of security, scalability, and deployability developers and operators expect from commercial-quality solutions.

Thus, the sequence of events leading to the 2003 electric power failures can be traced to a missing fault-tolerance and large-scale monitoring technology. This technical deficiency, in turn, is just a symptom of the broader market failure. A technology limitation unrelated to electric power grid management deprived the power industry of the tools it needed—and continues to need.

The story yields a basic insight. The economic forces in favor of restructuring the power grid far outweighed those that might have restrained the project for lack of certain tools. We restructured the grid anyhow but, without such tools, arrived at an unsatisfactory solution.

This is precisely what might now happen on a larger scale as Web services roll out into diverse critical settings. For example, the economic forces in favor of electronic health records might well trump any technical objections that we don't have the technology to do so in a trustworthy way.

Only replication can ensure access to critical data in the event of a fault.

WE HAVE WORK TO DO

What's the connection between trustworthy computing and data replication? In the case of the power grid, the key is a trustworthy technology for monitoring very large systems. But a monitoring infrastructure is just a tool for collecting and securely replicating sensor data, enabling applications and human observers scattered throughout a very large area to share a situational status report and coordinate an appropriate and consistent response when disruptions occur. Similarly, many aspects of trust and security reduce to data replication.

Platform vendors realize this, which is why so many platforms have an internal data-replication mechanism. But these solutions aren't accessible to the general developer. And the problems general developers face are often harder than the versions solved by vendors in their products.

For example, even if the government decided to build tools to monitor and control the power grid, not nearly enough is known about the fundamental science of collecting management information on such a large scale, or of replicating it to the needed degree. Researchers would need to instrument a massive critical infrastructure at hundreds of thousands of locations, with fault diagnosis occurring automatically and in real time, and all of this secured against intrusion or terrorist attack. This is considerably harder than, say, monitoring and managing the nodes in a computing cluster or a small data center.

Our inability to solve the large-scale problem is due to market forces, in several respects. Vendors aren't motivated to tackle the problem because customers aren't demanding solutions. But research funding for such efforts has suffered because DARPA, the NSF, and other major agencies are concerned that these kinds of investments often fail to transition into products. They tend not to invest in areas where research progress won't translate directly into better off-the-shelf solutions. Without the backing to explore robustness issues, researchers have moved to greener pastures.

To develop trustworthy computer systems, we must instill a wider social appreciation of the dimensions and limitations of trust in technology. And we must begin to take control of the implications of those limitations, rather than blindly building and deploying systems that simply can't rise to the standards of trust that their roles demand.

I'm an optimist, and I believe this is a challenge that we can actually solve. But doing so will take a concerted effort by both government and industry, starting with an earnest dialog on the issue of trust. It's past time for that debate to begin. ■

Ken Birman is a professor in the Department of Computer Science at Cornell University and a founding member of the Team for Research in Ubiquitous Secure Technology (TRUST; http://trust.eecs.berkeley.edu). His most recent book is Reliable Distributed Systems: Technologies, Web Services, and Applications *(Springer, 2005). Contact him at www.cs.cornell.edu/ken.*

Editor: Jack Cole, US Army Research Laboratory's Information Assurance Center, jack.cole@ieee.org; http://msstc.org/cole