Developing Secure Reliable Infrastructures for Web Services

Priya Narasimhan
Institute of Software Research International
School of Computer Science
Carnegie Mellon University, Pittsburgh, PA 15213-3891
priya@cs.cmu.edu

Web Services represent the next generation of enterprise infrastructures for the exchange of information and services across the Internet. The power of Web Services lies in their ability to abstract away platform- and language-specific details of service implementations, to expose service interfaces, or contracts, and to allow these self-contained self-describing services to be registered, published and discovered dynamically across the Internet. Exploiting the service contracts, developers can assemble powerful distributed Web-based applications using a combination of remote services, local services, and custom code. By using ubiquitous (HTTP, XML) and interoperable (SOAP) protocols, the client for a Web Service can be written in any language, and can run on any platform.

These advantages allow for simplified application complexity, and savings in terms of development and deployment time and cost. The dynamic grid of Web Services facilitates the rapid deployment of evolving enterprise interfaces and offerings. With this rich set of capabilities, along with protocols like SOAP, Web Services promise to be the "glue" for business-to-business enterprise interactions of the future.

However, the Web Services of today do not as yet incorporate QoS system properties, or "-ilities", such as reliability and security. Furthermore, the Internet, in its role as the underlying communication medium for Web Service interactions, is inherently unreliable, and has been known to exhibit system failures due to crashes and security attacks. In fact, HTTP is both the greatest strength and the greatest weakness (from a reliability perspective) of Web Services. Enterprises are already starting to deploy Web Services, even without sufficient security and reliability. With downtime being prohibitive in terms of cost and loss of reputation, and with faults and security breaches becoming increasingly unacceptable, the serious and widespread deployment of Web Services, in the absence of security and reliability guarantees, will not be realized.

This research exploits Web Services as the vehicle for exploring the *composition of reliability and security for wide-area dynamic Internet-based systems*. The focus in on understanding the *trade-offs* involved in composing diverse system properties (such as security and reliability), and at identifying any fundamental security or reliability concerns in today's Web Services.

Challenges

Web Services are intrinsically dynamic in nature because they represent an enterprise's product/service offerings over the web. In contrast to traditional client-server applications, the clients of Web Services can compose a number of Web Services (hosted potentially by different enterprises across the Internet) to form their own applications. No guarantees can be made about the dependability of the resulting application consisting of multiple Web Services, each with its own degree of reliability and security. The unreliability of any one of the Web Services used by this application could lead to the downfall of all of the others, even if they happen to be reliable. Thus, because these Web Service-based applications can straddle multiple enterprises, we must deal with, and reconcile, the reliability and security mechanisms/infrastructures/policies/guarantees of each individual enterprise involved in the application.

Upgrades are natural occurrences in the life cycle of any software. The composable nature of Web Services can cause the upgrade of an enterprise's Web Service to have far-reaching (and possible adverse) impacts across the Internet. Upgrading a Web Service requires reliable secure mechanisms, across the wide-area Internet, for client/user tracking, service uptime and usage monitoring, management of distributed state, change propagation to remote sites, coordination of resulting changes and side-effects to other Web Services, etc. This dynamic element ($i.e.$, runtime discovery and late binding, continuously evolving interfaces and offerings) of Web Services make the problems of dependability challenging, and far more interesting than conventional client-server enterprise systems.

Because Web Services are aimed at eliminating the boundaries of typical enterprise computing by opening up

enterprises to each other, it is possible for enterprises to invoke, and interact with, Web Services outside their firewalls. For instance, the client-side composition of multiple Web Services typically occurs outside the firewalls of enterprises. Thus, even if a single Web Service might be secure within its own firewall, there are no such guarantees about a collection of Web Services used by an application. Thus, the very potential of Web Services also makes them more vulnerable from a security standpoint.

Providing dependability in the face of an unbounded asynchronous communication medium like the Internet, is a challenge. There are distinct technical problems: (i) to provide availability so that there are always multiple copies, or replicas, of a Web Service that can provide the same functionality, with adequate response time, and (ii) to keep these replicas consistent in state so that there is no loss of data if any one of these replicas fails, (iii) to know what has occurred, what is yet to occur, and what parts of the system have been modified, in the event of a fault, (iv) to ensure that critical parts of the system are not compromised in the event of a malicious attack, (v) to ensure the reconciliation of diverse security mechanisms and policies of the constituent Web Services within an application. Above all of these, there is an over-arching goal to ensure that the client of the Web Service is not exposed to, and does not have to deal with, the security and reliability problems; thus, the client has to continue to perceive continuous uptime of the Web Service, and to be shielded transparently from the occurrence of faults or attacks.

Currently, the only attempts to make Web Services secure and reliable involve modifying the SOAP headers to carry additional information. For Web Services to be truly secure and reliable, there need to exist additional mechanisms that guarantee the

- Reliable secure messaging across the Internet, even as servers crash and as sites become disconnected
- Management of a Web Service's state to protect the application from loss of either data or processing, in the event of both faults or service upgrades, and
- Secure composition, outside the enterprise firewall, of remote Web Services,
- End-to-end reliability of an application that is composed of a grid of multiple remote Web Services, each with its own degree of reliability,
- Both reliability and security, even as the implementations and interfaces of an application's constituent Web Services are dynamically evolving, and
- Identification, and resolution, of the trade-offs that occur between and security and reliability themselves.

Combining Reliability and Security

Building infrastructures that must provide both reliability and security simultaneously to the application poses significant research problems. In most cases, from the perspectives of both the application and the infrastructure, security and reliability do not always make a good "marriage" [1].

Reliability necessitates replicating entities, and distributing the replicas across distinct processors, in order to avoid a single point of failure. However, the physical distribution of objects or components merely aggravates the security problems within the system: as the number of replicas of a component increases, the reliability of the component increases, but so does the number of ways in which the integrity or confidentiality of the data can be compromised. Also, the more distributed the replicas, the more reliable a system becomes, but the more vulnerable it is to security attacks. Reconciling such security $vs.$ reliability conflicts is non-trivial because it involves an in-depth understand of system-level issues, security and reliability principles, and the implications on end-to-end application behavior.

One strategy is to fragment the data or state of a Web Service in reasonable ways, and to scatter the fragments redundantly across a number of replicas. The idea is that each replica, even if compromised, contains only a small piece of the information and, by itself, cannot be used to deduce the entire state of the system. This becomes more challenging as Web Services are dynamically changing in state, functionality and behavior, and as they are used as remote runtime pieces of a larger application.

The interesting questions worth answering are: (i) in how many different ways can a service be fragmented so that each "slice" does not reveal too much if compromised, (ii) what is the minimum number of such "slices" which, when compromised, can be used to deduce the entire service, (iii) how many replicas of each "slice" should we employ, (iv) how do we detect, and handle, the situation where a replica has been maliciously compromised, and then starts to diverge in state and in behavior from the other copies of the "slice", and (v) how do the "slices" of one Web Service combine with those of others within an application?

Conclusion

This research exploits Web Services for exploring the composition of reliability and security for wide-area dynamic Internet-based systems, with the aim of identifying and resolving the trade-offs involved in composing diverse system properties (such as security and reliability). Web Services are particularly interesting because they are Internet-based systems that are dynamic, constantly evolving, and involve the composition of multiple remote services within a single application.

[1] This is evident from the specifications for Fault-Tolerant CORBA and CORBA's Security Service - both were independently developed, and are effective on their own. However, it is simply not possible to build a secure fault-tolerant CORBA system through the straightforward combination of both specifications.