Chapter 5
Forensics over Web Services:
The FWS

Murat Gunestas
General Directorate of Security, Ankara, Turkey

Duminda Wijesekera
CS Department, George Mason University, USA

Anoop Singhal
National Institute of Standards and Technology, USA

ABSTRACT
Web services are currently a preferred way to architect and provide complex services. This complexity arises due to the composition of new services by choreographing, orchestrating and dynamically invoking existing services. These compositions create service inter-dependencies that can be misused for monetary or other gains. When a misuse is reported, investigators have to navigate through a collection of logs to recreate the attack. In order to facilitate that task, the authors propose creating forensic web services (FWS), a specialized web service that when used would securely maintain transactional records between other web services. These secure records can be re-linked to reproduce the transactional history by an independent agency. Although their work is ongoing, they show the necessary components of a forensic framework for web services and its success through a case study.

INTRODUCTION
Web services are becoming a popular application of SOA (Service Oriented Architecture) within organizations, being used for many financial, government and military purposes. They do so by seamlessly integrating web services of different organizations over the Internet using choreographies, orchestrations, dynamic invocations, brokers etc.; and are now extending their way to include transactions that involve more than two participants, a.k.a. multiparty activities. These service-level compositional techniques create complex inter-dependencies between web services belonging to different organizations that can be exploited due to some localized or compositional flaws. Therefore, such exploits/attacks (Vorobiev & Jun, 2006; Demchenko, Gommans, & Oudenaarde, 2005; Singhal, Winograd, & Scarfone, 2006) can affect multiple servers and organizations, resulting in financial loss or infrastructural damage. Investigating such incidents would require that dependencies between service invocations be retained.
in a participating party neutral and secure way so that the alleged activity can be undeniably recreated while preserving evidence that could lead to and support appropriate prosecutorial activity. Material evidence currently extractable from web servers such as log records, XML firewall alerts from end point services, and the like, do not have any forensic value because defendants can rightfully claim that they did not send that message, and plaintiffs can fabricate or alter the log record to deceive the court. We describe a participant neutral, non-refutable solution, a forensically valid evidence gathering mechanism for SOA, through this chapter.

**BACKGROUND**

Two conceptual elements base current web services: (1) Use of XML (eXtensible Markup Language), SOAP (Simple Object Access Protocol), and WSDL (Web Service Definition Language) as basic building material; (2) Complex applications built upon long-running, sometimes transactional executions created from basic elements using choreography, orchestration and compositional methods.

**Basic Paradigm**

XML format underlies entire web service architecture and its artifacts. All schemas, definition files, and messages transmitted are formed by the means of XML. WSDL, a XML based definition file, defines the interface of a web service in order for the service to be invoked by other services in accordance with the specifications of internal executions. SOAP, a XML based protocol, defines the metadata of the messages to be exchanged between services. Operations are defined in WSDL documents and they are the only mechanisms that can be employed for web services to communicate with each other. SOAP messages are defined and exchanged as incoming and outgoing messages through the operations. WSDL proposes four types of operations:

- **Notification**: One message is sent to many receivers, such as broadcasting.
- **One-Way**: The message is sent and no response is expected, such as Fire-and-Forget.
- **Request-Response**: A typical RPC structure: The message is sent from sender to receiver and response is pushed back to the sender.
- **Solicit-Response**: Request is sent without any data and the response is expected.

Although there are four proposed operation types, the message exchanges can be defined in two ways, in summary, One-Way and Request-Response—this is so since notification and solicit-response can both be represented by one-way and request-response types, respectively.

**Composition Paradigm**

The message exchange patterns (MEP) described above form the base for the entire web service paradigm. These simple MEPs construct collaboration scenarios using the appropriate composition models. While defining a composition, two issues arise: first, how it is designed and second, what pattern it employs.

**Design Types**

Selecting the target provider services can be accomplished either statically or dynamically, that is, in design-time or run-time. Design-time selections entail a-priori determination while run-time selections can introduce the opportunity to switch between web services among selected domains.

**Static Composition**: Static compositions propose web services to be selected and determined through the business applications at design-time. Currently, most web service implementations are
Related Content

Security and Privacy Requirements Engineering
www.igi-global.com/chapter/security-and-privacy-requirements-engineering/123534?camid=4v1a

Planning for Hurricane Isaac using Probability Theory in a Linear Programming Model
www.igi-global.com/article/planning-hurricane-isaac-using-probability/76657?camid=4v1a

Cyber Defense Maturity Levels and Threat Models for Smart Cities
www.igi-global.com/article/cyber-defense-maturity-levels-and-threat-models-for-smart-cities/226948?camid=4v1a

Do You Know Where Your Data Is?: A Study of the Effect of Enforcement Strategies on Privacy Policies
www.igi-global.com/chapter/you-know-your-data/49513?camid=4v1a