Misuse Pattern: Spoofing Web Services

Jaime Muñoz-Arteaga
Univ. Autónoma de Aguascalientes
Avenida Universidad #940. C.P. 20131, Aguascalientes, México
jmunozar@correo.uaa.mx

Héctor Caudel-García
Univ. Autónoma de Aguascalientes
Avenida Universidad #940. C.P. 20131, Aguascalientes, México
kaudel@gmail.com

Eduardo B. Fernandez
Florida Atlantic University,
Boca Raton, FL 33431, USA
ed@cse.fau.edu

ABSTRACT
A misuse pattern describes how a misuse is performed from the point of view of the attacker, what system units it uses and how, provides ways of stopping the attack by enumerating possible security patterns that can be applied for this purpose, and provides forensic information. This pattern is useful for designers and developers of web services, who can then avoid these situations following the prescriptions of the pattern. This pattern could also be a guide to know what happened and to correct the corresponding vulnerabilities that led to the attack. We present here a misuse pattern, Spoofing Web Services. A web service spoofing misuse tries to impersonate the identity of a user, and then with the user´s credentials makes requests in his name, with the intention of accessing a specific web service.

Keywords
Authentication, Misuse Patterns, Security, Spoofing attacks, Web Services.

1. INTRODUCTION
The World Wide Web Consortium defines a web service as “a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically Web Services Description Language WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards." [1]. Web services can communicate and share information with other web services or applications. However, there are some web services which contain sensitive information targeted to a particular set of users. In that case, the resources from some web services can only be accessed by authorized users, who must be first authenticated. Authentication is a fundamental security mechanism to establish the identity of users, software components or entire systems. It determines if someone or something is who it claims to be.

Many of the features that make web services attractive, including greater accessibility to the data, dynamic application-to-application communication, and relative autonomy (lack of human intervention) may bring serious security challenges. Security has been an important quality factor for web services including in particular authentication. Some of the problems that web services could have with respect to authentication include: Spoofing, Forged Claims and Man-in-the-Middle. In particular, spoofing refers to tricking or deceiving computer systems or other computer users by hiding one's identity or faking the identity of another web service.

This work presents a description of spoofing web services in the form of a new type of pattern, a misuse pattern [3,4]. A pattern is a well understood solution to a recurrent problem in software design and summarizes best practices about its topic [2]. Misuse Patterns describe from the point of view of the attacker, how a type of attack is performed (what system units it uses and how) proposes ways of stopping the attack by enumerating possible security patterns that can be applied for this purpose, and helps analyze the attack once it has happened by indicating where we can find forensics data as well as what type of data. We have published several misuse patterns for different types of attacks [4,12].

We present here a misuse pattern, Spoofing Web Services, which describes how to take a false identity when accessing a web service. This pattern is useful for the designers and developers of web services, who can then avoid these situations following the prescriptions of the pattern. This pattern can also be a guide to know what happened after an attack and shows how to correct the corresponding vulnerabilities that lead to the attack. Additionally, a misuse pattern provides an insight into how an attack is carried out, as well as the extent to which the attack affects a web service (access to resources, loss of trust, or others).

The paper is organized as follows: Since misuse patterns are not widely known, we present first a template used to describe these patterns in Section 2. Section 3 describes a misuse pattern for Spoofing Web Services. Section 4 presents some conclusions.

2. A TEMPLATE FOR MISUSE PATTERNS
Name

The name of the pattern should correspond to the generic name given to the specific type of attack in standard attack repositories such as CERT [5].

Intent

A short description of the intended purpose of the pattern, i.e., which problem it solves for the attacker.

Context

The context describes the generic environment including the conditions under which the attack may occur. This may include minimal defenses present in the system as well as typical vulnerabilities of the system. The context can be specified using a deployment diagram of the relevant portions of the system as well as sequence or collaboration diagrams that show the normal use of

the system. A class diagram may show the relevant system structure. We can list specific preconditions for an attack to happen.

Problem

From an attacker's perspective, the problem is how to find a way to attack the system. An additional problem occurs whenever a system is protected by some defense mechanisms. The forces indicate what factors may be required in order to accomplish the attack and in what way; for example, which vulnerabilities can be exploited. Also, which factors may obstruct or delay accomplishing the attack.

Solution

This section describes the solution of the hacker's problem, i.e., how the attack can reach its objectives and the expected results of the attack. UML class diagrams show the system under attack. Sequence or collaboration diagrams show the exchange of messages needed to accomplish the attack. State or activity diagrams may add further detail.

Affected system components (Where to look for evidence)

This is a new section compared to standard security patterns. The misuse pattern should not be a comprehensive representation of all components and relationships involved in an attack. Rather, the pattern should represent all components that are important to prevent the attack and are essential to the forensic examination. This can be represented by a class diagram that is a subset or superset of the class diagram of the context.

Known uses

Specific incidents where this attack occurred are preferred but for new vulnerabilities, where an attack has not yet occurred, specific contexts or scenarios where the potential attack may occur are enough.

Consequences

Discusses the benefits and drawbacks of a misuse pattern from the attacker's viewpoint. Is the effort and cost of the attack commensurate with the results obtained? This is an evaluation that must be made by the attacker when deciding to perform the attack; the designers should evaluate their assets using some risk analysis approach. The enumeration includes good and bad aspects and should match the forces.

Countermeasures and Forensics

This section describes the security measures necessary in order to stop, mitigate, or trace this type of attack. This implies an enumeration of which security patterns are effective against this attack. From a forensic viewpoint, it describes what information can be obtained at each stage tracing back the attack and what can be deduced from this data in order to identify this specific attack. Finally, it may indicate what additional information should be collected at the involved units to improve forensic analysis

Related Patterns

Discusses other misuse patterns with different objectives but performed in a similar way or with similar objectives but performed in a different way.

3. MISUSE PATTERN: SPOOFING WEB SERVICES

Name

Spoofing Web Services.

AKA Principal Spoofing in web services

Intent

A web service spoofing misuse tries to impersonate the identity of a user by stealing his credentials, and then makes requests in his name with these credentials, with the intention of accessing a victim's web service.

Context

Through use of web services, enterprises can exchange data. In order to let other entities access the web service they publish a WSDL file with functionalities and security policies.

The client must create a request which should adapt its methods and policies to those of the web service provider. Security can be implemented at the level of messages, each user is given credentials to access services and the responsibility of protecting his messages from other users. The web service can be protected using standards such as WS-Security [6] to protect messages and WS-Policy [8] to verify the security requirements of the web service.

Problem

How we can effectively perform a Principal Spoofing attack against web services so we can access the information of another user?

The attack can take advantage of the following **vulnerabilities**:

- Currently there is no method to verify identity unequivocally.
- When the WSDL file is published, the interfaces of the web service are exposed, and thus its entry points.
- The source address of the request can be altered.
- There is a tendency to trust the presented credentials, without a more thorough check.
- Once the authentication stage is passed, other attacks can be generated more easily, for example, DDoS attacks
- Maybe the interchanged data items are not encrypted, and it may be possible to steal the credentials using special software.
- The resulting damage depends on the privileges of the credentials submitted.

Solution

The security standards use the credentials of the user to protect communication between web services, by signing and encrypting messages. If the attacker manages to get valid credentials, he can communicate with other web services. This type of vulnerability is mentioned in [9] and is named "Principal Spoofing". In this attack, the user does not know that his credentials were stolen, until the damage is done (such as alteration of information, access to his resources, alteration of privilege, even attaching malicious code that could be harmful to the server).

When the attacker has the credentials, he can use the WSDL (Web Service Definition Language) file in order to discover the security policies of the web services and create a valid message. This situation is conditional on the attacker obtaining a user´s credentials, the valid user not knowing of the theft and not having reported the theft.

This attack can be carried out, taking into account the following aspects:

1. If the communications between web services are not protected, sniffer software such as WireShark or WebScarab, could intercept packages while travelling in the network, and obtain the credentials carried in the messages.
2. Using social engineering to get the credentials of the user.
3. The system could be using inadequate security policies; it is possible to cheat the system in this way.
4. Another vulnerability may be present if the system does not verify the origin of the request.

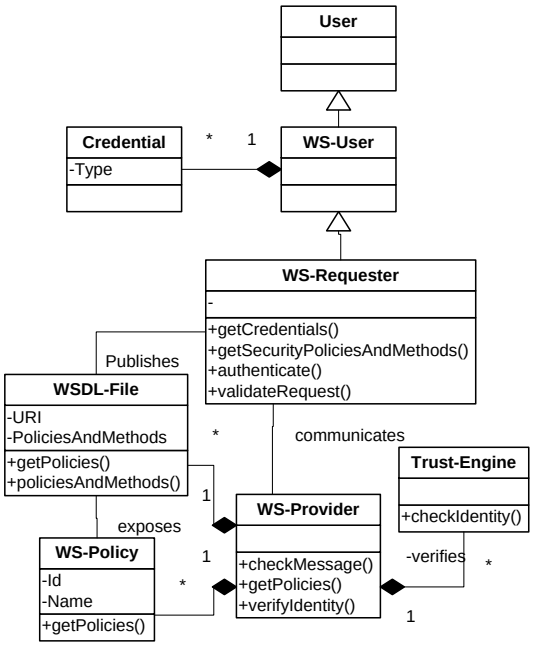

Figure 1: Class Diagram for Web services spoofing

The class diagram of Figure 1 shows the typical relationships between web services. The *WS-User* has *Credentials* to communicate with other web services. The *WS-Requester* can be a *WS-User* o another kind of web service. The *WS-Provider* has several *WS-Policies*, and these are stored in the *WSDL-File*. In this way, a *WS-Requester* can be created, based on the *WS-Policies* shown in the *WSDL*. When the *WS-Requester* tries to communicate with the *WS-Provider*, the *WS-Provider* checks the request with its *Trust Engine*, which checks the requester's credentials**.**

To make possible the Principal Spoofing attack, the user must have the user´s credentials as shown in *Figure 2*. This diagram shows the structure of a Principal Spoofing Attack, the classes *WS-Attacker* and *Attacker* being the new elements introduced by the attacker.

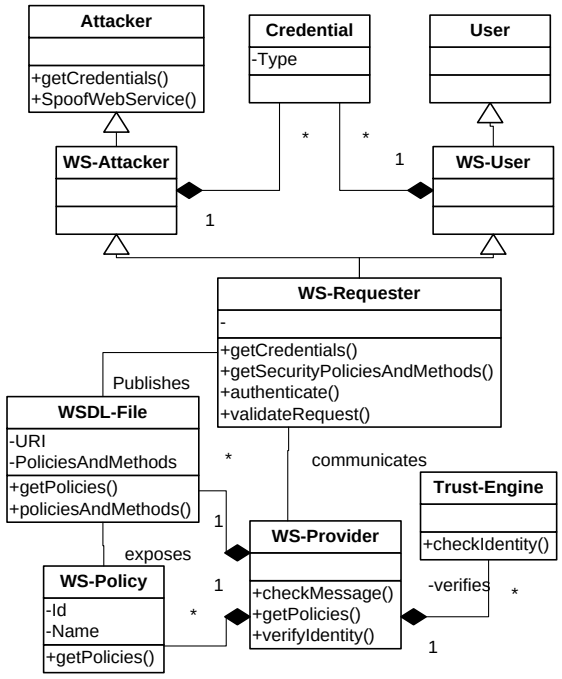

Figure 2: Misuse Pattern: Spoofing Principal

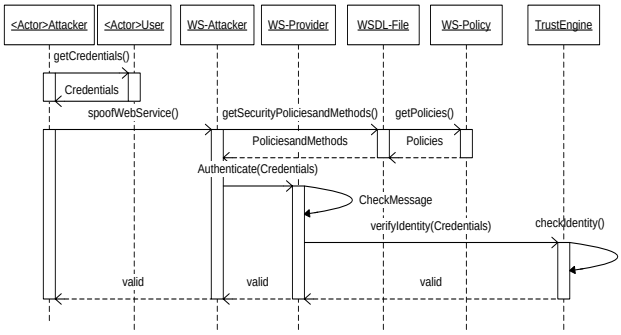

Figure 3: Principal Spoofing Sequence Diagram

The sequence diagram of Figure 3 shows the sequence of steps to perform a Principal Spoofing Attack. In the way indicated earlier, an *Attacker* can obtain credentials of a valid *User*. When the attacker has the credentials, he can access the *WSDL-File* of the *WS-Provider*, where the *WS-Policies* and methods to access the web service are published. The *WSDL-File* can be found through the UDDI service or it can be accessed through techniques such as "crawling". With the information obtained from the *WSDL-File* and the credentials of the *User*, the *Attacker* can create a web service.

When the attacker has the web service, he can make a request to the *WS-Provider* using the credentials of the user. The *WS-Provider* receives the request and checks the message against the policies of the web service. If the message is correct, the WS-

Provider, sends the credentials to the *Trust Engine* to check the identity of the user. Because the credentials are valid, the request will be authenticated. When security policies are known, the attacker will be able to develop another web service with a different behaviour, according to his intentions. User credentials will be used as identification to access the system´s functionality.

Known uses

Scenario I

Hole [10] describes some online banking security issues identified in a period of two-years. During this period, several attacks were identified against the observed bank system by combining simple brute-force attacks with Distributed-Denial-of-Service (DDos) attacks that exploit a bank login procedure. In the bank, the authentication procedure was based on a web service, which was the only point for access to users' information and which authenticated requests.

A web service could be deployed in the bank or externally. This web service using IP Spoofing became a victim of a DDoS attack because it was not well configured; all the requests that it received were sent to the main server, skipping the authentication in the web service from the bank. With this attack, the bandwidth and the resources of the bank are affected; in the worst case, the system can go down. This could be done if the attacker had the credentials of a user from the bank.

Scenario II

In this case a web service from an information service provided information only to authorized personnel. Every user had credentials to use the web service, needed because of the sensitive information being handled. The attacker obtained credentials from some users and built another web service to try to access the information, but the real damage was more than just obtaining the confidential information.

The attacker was also able to make several requests to the web service target, using the credentials from a valid user, the requests being generated from several parts of the network. The attacker used techniques from IP Spoofing to hide his position. The web service target can verify first the identity from sender of the message or check the structure of the message and finally check the identity of the requester. With this scenario, several attacks can be realized [2]. All of these were DDoS attacks.

Scenario III

In October 2010, a Firefox plug-in, "Firesheep", was delivered. Through this plug-in, an attacker without much technical knowledge, could access the information of other users in the same wireless network. In this way, the attacker could see in his browser the information exchanged by other sites, some as important as Facebook, Twitter, or Google. With a small modification to this plug-in, it is possible to build a sniffer. When a session is captured, the credentials of the users could be captured without their knowledge.

The first of these attacks uses *Resource Exhaustion*. Such types of attack target at compromising a web service´s availability by exhausting the resources of the service´s host system, such as memory, processing resources, or network bandwidth. With a technique named Oversize Payload, the attack can be performed using a large SOAP message document.

Another attack named Attack Obfuscation is made using the WS-Security standard. By providing confidentiality to sensitive data, XML Encryption can hide the contents of messages. The encrypted message can contain an intended attack, like Oversize Payload, XML Injection or another. The advantage of this method is that the defender must decrypt the message to be able to analyze it.

The attacks described above, are based on the trust that the system may have on its users, because an important step in the attack is the verification of the identity of the requester.

Consequences

The success of this attack implies:

- The attacker can access the sensitive information from the valid user
- Malicious code can be attached to the messages to bring further attacks on the user or his associates.
- The attacker can read or modify information or make transactions in the name of the valid user.
- Knowing the attacker's behavior, forensics proofs could be shown in order to verify that the attack was carried out, and by whom.

Possible disadvantages include:

- If an attacker doe not know about the forensics evidence that he would generate, he might be identified.
- Some defences described in the next section can stop this attack.

Affected Components

When the attack is finished, several components will be affected:

- *WSDL-File:* This file will not be trustable anymore because it was used by the attacker, and this file, will contain the vulnerabilities of the web service.
- *User Credentials:* The user´s credentials, are compromised, so they are not useful anymore and should be revoked.
- *User Trust:* The user trust in the authentication method will be compromised.
- *WS-Requester:* When the attack is finished, the requests from the authorized web service now are suspicious.

Countermeasures and Forensics

Human behaviour cannot be controlled, so we cannot avoid that humans be deceived into releasing their credentials. Also, the victim can be robbed of his credentials.

The attack can be mitigated through the following measures:

1. Encrypt communications between web services. Encryption can be performed by following security standards such as XML Encryption and WS Security.
2. For each of the web services that are developed:
    a. Generate an internal credential, which can be an Id or some kind of token. This credential will be embedded in the code.
    b. The web service provider must publish a policy about credential requests for each transaction.
    c. Also, if the communication is done using a PKI Infrastructure, the credential must be sent encrypted using the public key provided by the web services

provider. With this measure even if the message is intercepted by sniffer software, the attacker will not be able to access the content of the message.

Likewise, the following forensic evidences are possible:
--With logs on the web service provider we can identify which user was affected, and what were the actions performed.
--Check the procedence from the attack (IP address), although the attacker can spoof its IP address.

Related Pattern
WS-Trust**:** This pattern helps to establish the relationships that should arise between web services, in order to share their credentials [6].

4. CONCLUSIONS

Authentication is an important concern for web services and misuse patterns can help to analyze their attacks and evaluate the security of a complete system. This paper has presented a misuse pattern, Spoofing Web Services, which describes how this attack is performed and how to stop it. A catalog of misuse patterns can be very useful for designers and users, we have built one for VoIP attacks [12], but we need catalogs for other aspects of the system.

ACKNOWLEDMENTS

We thank our shepherd, Foutse Khomh, for his valuable comments that improved our paper. We also thank CONACYT and the Universidad Autonoma of Aguascalientes for their support for the summer stay of Hector Caudel at Florida Atlantic University.

REFERENCES

[1] World Wide Web Consortium (W3C) , http://www.w3.org, access date: September 20, 2010.

[2] E. Gamma, R. Helm, R. Johnson, and J. Vlissides (1995). Design Patterns: Elements of Reusable Object-Oriented |Software. Addison-Wesley.

[3] E. B. Fernandez, N. Yoshioka, and H. Washizaki, "Modeling Misuse Patterns," International Conference on Availability, Reliability and Security, 2009. ARES '09, pp. 566-571, Mar. 2009.

[4] E. B. Fernandez, J. C. Pelaez, and M. Larrondo- Petrie, "Attack Patterns, A New Forensic And Design Tool," in IFIP International Federation for Information Processing, P. Craiger and S. Shenoi, Eds. Springer, Boston, 2007, vol. 247, Ch. 24, pp. 345-357.

[5] CERT Coordination Center. Carnegie Mellon University. http://www.cert.org/, Access date: January 15, 2011.

[6] O. Ajaj and E. B. Fernandez, "A pattern for the WS-Trust standard for web services," in AsianPLoP 2010: 1st Asian Conference on Pattern Languages of Programs, Tokyo, Japan, 2010.

[7] L. J. Camp, Identity, Authentication, and Identifiers in Digital Government, in International Symposium on Technology and Society, 2003, pp. 10-13.

[8] A. Barbir, C. Hobbs, E. Bertino, F. Hirsch, and L. Martino, "Challenges of testing Web Services and security in SOA Implementations," in Test Analysis of Web Services. SpringerLink, Sep. 2007, pp. 395-440.

[9] P. Morrison and E.B.Fernandez, "The Credential pattern", Procs. of the Conference on Pattern Languages of Programs, PLoP 2006, Portland, OR, October 2006, http://hillside.net/plop/2006/

[10] K. J. Hole, V.Moen, T. Tjostheim, "Case study: online banking security," Security & Privacy, IEEE , vol.4, no.2, pp.14-20, March-April 2006

[11] M. Jensen, N. Gruschka, R. Herkenhoner, N.Luttenberger, "SOA and Web Services: New Technologies, New Standards - New Attacks," Web Services, 2007. ECOWS '07. Fifth European Conference on , vol., no., pp.35-44, 26-28 Nov. 2007

J. Pelaez, E.B.Fernandez, and M.M. Larrondo-Petrie, "Misuse patterns in VoIP", Security and Communication Networks Journal. Wiley, vol. 2, No 2, 635-653, published online**:** 15 Apr 2009, http://www3.interscience.wiley.com/journal/122324463/abstract