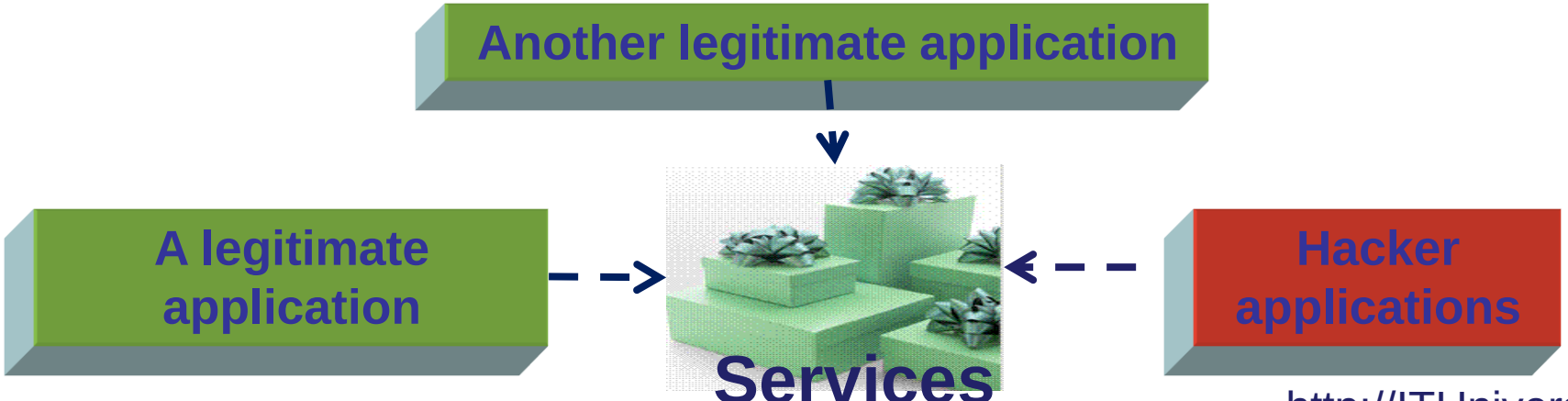
SOA and Web Service Security
A Problem

Service-Oriented Architecture shift development focus from applications to services. Multiple applications can call the same services instead of copy/paste/modify their code.

The problem is that exposed services can be called not only by legitimate applications. Being outside of application umbrella, exposed services need secure protection.



SOA and Web Service Security Solution

What:

Service request must include:

- Secure identification of an application requested a service

- Prove that the application has the proper access rights

- Prove that the data are protected and have not been changed

How:

Multiple layers of security provide better security.

Using SSL over HTTP we ensure that all messages are encrypted

This means that web users will access applications with the URL that starts with **HTTPS:** and served by the SSL port (usually 443)

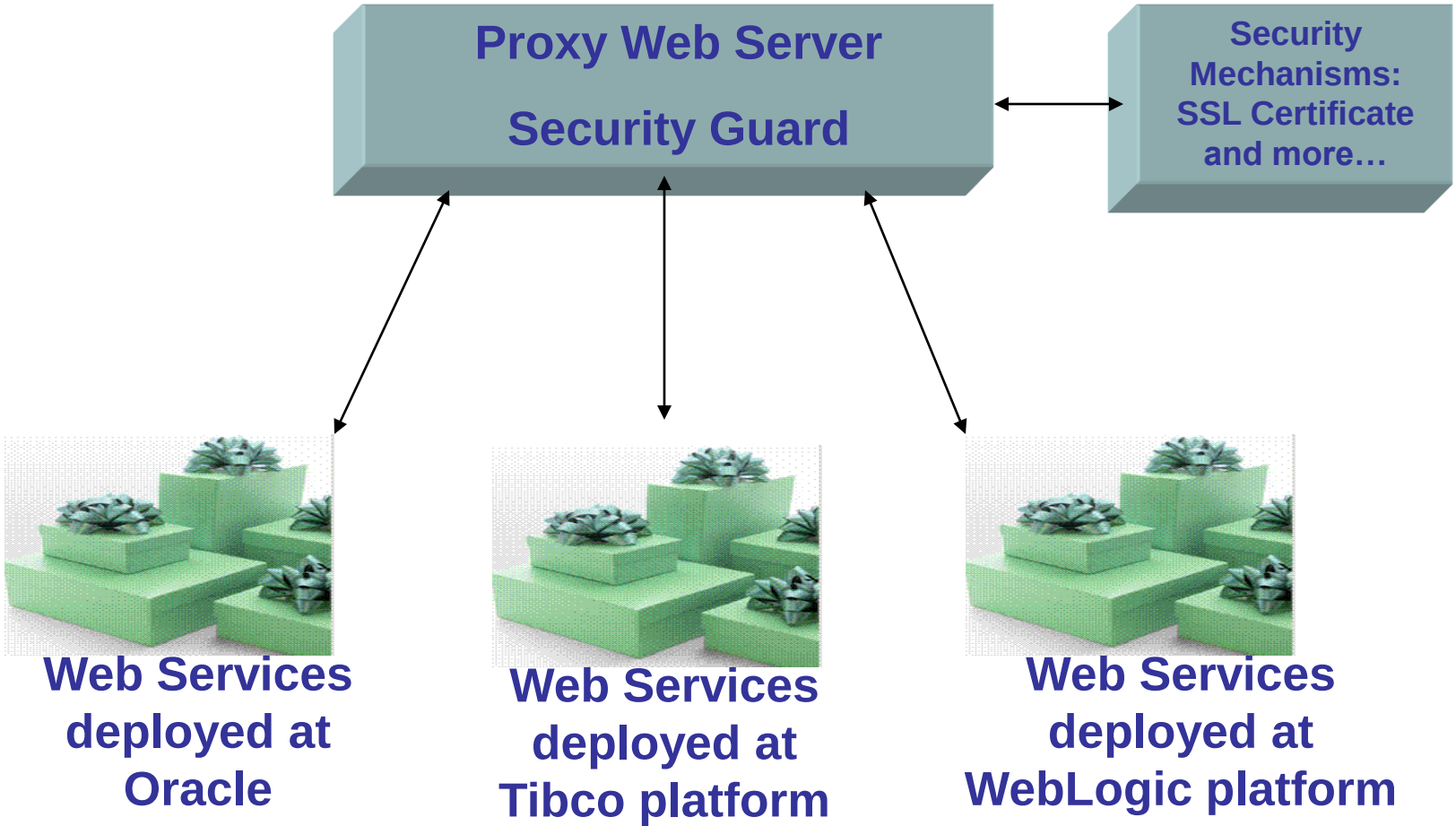
Another layer is to protect users from authentication fraud by establishing rules for password encryption and change password functionality.

Working in Java environment, it is recommended to use Java encryption library and proven encryption mechanisms, versus homegrown encryption algorithms.

Establish a single Security Guard protecting services deployed at multiple locations.

Web Service Security Guard

A proxy web server is a single point of access for multiple internal and external consumers accessing multiple web services



How to access web services from Java applications

1. **Use the "keytool" (from the java/bin directory) to import a certificate (for example, itsint_cert1.cer) into the Java keystore**

prompt>keytool -import -noprompt -keystore C:\java\jdk\jre\lib\security\cacerts -storepass changeit -file c:\temp\itswsint_cert1.cer -alias itsws

2. Include the HTTPS URL in the command line of a Java application (for example, starting the batch file)

prompt>authclientXFire.bat https://tst.javaschool.com/AuthService getRoles ad testuser2

If Java or .Net applications need to access a web service which requires HTTP Basic Authentication, the application should send a username and password in the HTTP header of the request. Every request, including the first, MUST contain the **basic authentication data**.

The username and password are base-64 encoded in the **Authorization HTTP header** in this format: Basic <username>:<password>. See [RFC 1945] section 11.1 for more information

Authentication Rules and Tokens

For **application-based** transactions, the following authentication mechanisms is recommended:

1. **SSL Client (Mutual) Authentication**

For **user-based** transactions in which the Service Consumer is being used by a person, the following additional authentication mechanisms are recommended:

2. HTTP Basic Authentication

3. WS-Security **UsernameToken** to pass the username and password
Or **BinarySecurityToken** as Kerberos Tickets or X.509 Certifications

Passing Tokens in the WS-Security SOAP Header

Examples of the UsernameToken and BinarySecurityToken in the SOAP Headers

```
<wsse:UsernameToken>
  <wsse:Username>{userName}</wsse:Username>
  <wsse:Password Type="wsse:PasswordDigest">{encryptedPassword}
</wsse:Password>
  <wsse:Nonce>{encryptedSiteID}</wsse:Nonce>
  <wsu:Created xmlns:wsu="http://schemas.xmlsoap.org/ws/2002/07/utility">
2013-11-20T00:44:02Z</wsu:Created>
</wsse:UsernameToken>
```

*The **Nonce** field is used to validate that the contents came from the current site
***wsu:Created** field indicates when a particular element was added to the message

```
<wsse:BinarySecurityToken ValueType="wsse:X509v3"
EncodingType="wsse:Base64Binary" Id="“{SecurityTokenID}">{theToken}
</wsse:BinarySecurityToken>
```

Reference: WS-Security Specification | WS-Security Addendum | http://ITUniversity.us

Security Assertion Markup Language (SAML)

SAML is an **XML standard by OASIS** to securely exchange user authentication and authorization data between web domains.

Allows for **web browser single sign-on (SSO)**

Defines three roles:
- The **principal** (typically a **user**),
- The **identity provider** (**IdP**),
- The **service provider** (**SP**)

<saml:Assertion Example>
Assertion *A* was issued at time *t* by issuer *R* regarding subject *S* provided conditions *C* are valid
</saml:Assertion>

Example of Using SAML for Google Apps, Where Google plays the role of a Service Provider