Securing Multi-Tiered Web Applications

George Mathew, Xiaojiang Du
Department of Computer and Information Sciences
Temple University
Philadelphia, PA, 19122, USA
Email: {George.Mathew, dux}@temple.edu

***Abstract* - Multi-tiered architecture is very common in today's enterprise web applications. It is necessary to secure channels in each tier in order to secure a multi-tiered web application. For a non-HTTP based channel, there are several options to secure the channel. These security options have been used in a number of applications. However, it is not clear which option has better performance (such as delay, security strength, etc). In our research, we conducted real-network experiments to study the performances of several popular security protocols that are being used for securing multi-tiered web applications. Our experimental results provide several useful insights and guidelines for the design and deployment of secure multi-tiered web application.**

Keywords - Security; web applications; multi-tiered applications

I. INTRODUCTION

Multi-tiered architecture is the norm in today's enterprise web applications. A typical 3-tier application has a user interface presentation layer, a business logic layer and a data access layer. For example, a web based banking system may have 3 tiers: *tier 1* - the user login forms and menus of transactions served by a web server; *tier 2* - the business logic layer served by an application server (e.g., Websphere or Weblogic) and doing all logistics of online transactions (e.g., electronic fund transfer to a different bank); and *tier 3* - a backend database (e.g., SQL Server or Oracle). For reasons of scalability and separation of concerns, it is a common practice to deploy each tier on a separate (dedicated) server. Multi-tiered web applications have a point-to-point communication channel at each tier. To achieve end-to-end security, each channel needs to be secured. This is due to the fact that encryption and decryption happens point-to-point as well as the fact that the protocols used in each tier may be different. Since powerful network analysis tools such as Wireshark [1] are easily (and freely) available, sniffing packets can be easily done. Hence, it is essential to encrypt packets in each tier.

In one of the early surveys on web security (in 1998), Rubin and Geer [2] identified server security, mobile code, data transfer, and user privacy as some of the particular areas of concern. Later (in 2005), McDaniel and Rubin [3] noted that "the investigation of Web security is in its infancy and much work remains". Web-based attacks account for 20-30% of all network attacks [4]. Anomaly detection [4] and trusted computing [5] have been proposed to improve trust in web transactions. The large and growing installation base of web sites makes them easy targets for eavesdropping and other cyber attacks.

In our work, we studied a 2-tier web application. However, our results apply to 3 (or more) tier applications. A 2-tier structure is illustrated in Fig. 1. The communication channel in tier 1 is from a client browser to the application server. The channel in tier 2 is between the application server and the backend directory server.

For multi-tiered web applications, it is common to have a login component in the first tier. This is usually in the form of a <username, password> tuple. The username and password are used to authenticate the user against some backend user profile (e.g., a relational database, a directory service or some other container). A successful authentication results in successful logging into the application system. The first tier web communications use HyperText Transfer Protocol (HTTP). Hypertext Transfer Protocol Secure (HTTPS) [6] is typical used to provide security for the first tier. HTTPS is a combination of the Hypertext Transfer Protocol with the Secure Sockets Layer (SSL) / Transport Layer Security (TLS) protocol and can provide both encryption and authentication. HTTPS is supported by most browsers (Internet Explorer, Firefox, Safari, Chrome, Opera, etc.) and web servers (Apache, IIS, Tomcat, Jetty, etc.). The standard ports for HTTP and HTTPS are 80 and 443, respectively.

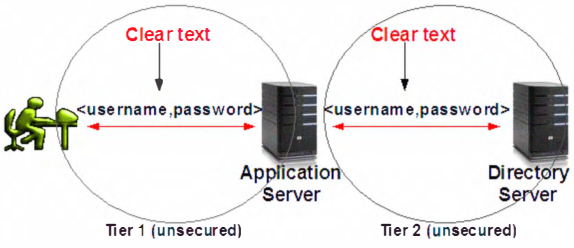

Figure 1: Two-tiered web application

However, the second tier connection to a container is not necessarily based on HTTP. For example, a relational

978-1-4244-5849-3/10/$26.00 ©2010 IEEE

database uses SQL-based protocols through proprietary or de facto standard protocols similar to ODBC. Another example is the Lightweight Directory Access Protocol (LDAP) [7], which uses standards-based API defined in RFC 1823 [8]. Since the second tier could be non-HTTP based protocols, there are several options to secure the second tier communications. It is interesting to study various options for protecting this tier irrespective of the underlying communication protocols. In this project, we study the performance of several protocols that may be used for securing tier 2.

II. SECURITY PROTOCOLS AND IMPLEMENTATION

In this section, we provide introductions of tier 2 protocols and their implementations in our project. LDAP is used for directory access without security protection. LDAPS enables LDAP over SSL and it is a secure version of LDAP. In addition, SSH [9] (Secure Shell) and IPSec [10] may be used to provide security for tier 2. We describe these protocols in subsections A, B, and C, respectively.

A. LDAP and LDAPS

LDAP has its roots in Directory Access Protocol (DAP) which was designed by International Telecommunication Union (ITU). The design of DAP was driven by the need for a global network based directory. This led to the X.519 standard for DAP. DAP was based on the OSI layers. The implementation of DAP was quite big and very resource intensive to run. The consequence of this was the birth of LDAP, which is much lightweight and based on TPC/IP. The initial development of LDAP was done at the University of Michigan. Later it was accepted as an IETF standard - RFC 1487 [11]. Regular LDAP communication uses TCP over port 389. The data is encoded in ASN.1 format while in transit between the server and client. This encoding can be decoded easily and the security level is as weak as plain text. LDAPS enables LDAP run over SSL and it uses port 636 (see Fig. 2). LDAPS requires configuring the LDAP server with certificates for secure communication.

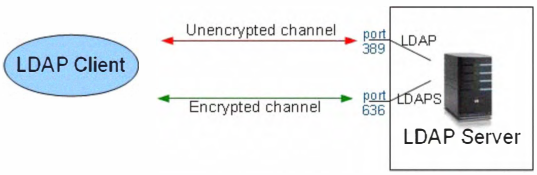

Figure 2: LDAP/LDAPS communications

LDAP is primarily used as a container for user information in an organization. One of the most widely used standard for user profile is a schema called inetOrgPerson, which is defined in RFC 2798 [12]. LDAP also provides a mechanism for authenticating users by username and password as recorded in the inetOrgPerson schema. This is a popular authentication mechanism supported in web browsers and other software. For example: Microsoft's Active Directory, CAS [13] (from Yale University) and Shibboleth [14] (from Inernet2) can use user credentials from LDAP to authenticate users.

B. SSH

SSH [9] is a popular security scheme. SSH is a set of utilities modeled after the 'r' utilities (rsh, rcp etc.) from the traditional UNIX environment. When using rcp, rlogin, rsh, telnet, etc., user passwords and other sensitive information are transmitted across the Internet unencrypted. SSH encrypts all traffic and can effectively defend eavesdropping, connection hijacking, and other attacks. Additionally, SSH provides secure tunneling capabilities and several authentication methods. The openssh [15] implementation supports all SSH protocol versions and capabilities. SSH port forwarding transparently encrypts an application's data stream. This transparency is achieved at the application layer and not at the network layer. SSH port forwarding can be used only for TCP.

In this project, we set up an SSH channel for tunneling LDAP traffic between the application server and the directory server. Fig. 3 shows the port redirection of the traffic. A local port 'xyz' is chose on the loopback interface (usually resolved by the hostname 'localhost' and has the IP address 127.0.0.1). A port redirection is set from 'xyz' to the port 389 on the LDAP server through the ssh tunnel. Once this redirection is in place, the client application will connect to port 'xyz' on the localhost and the port redirection will channel the packets back and forth through the SSH tunnel.

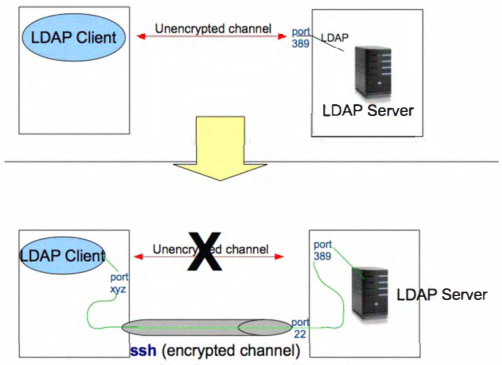

Figure 3: SSH port redirection

C. IPSec

IPSec is a set of protocols and has three major components [17]: the Internet Key Exchange (IKE), Authentication Header (AH) and Encapsulating Security Payload (ESP). IKE manages and distributes keys. IKE is also responsible for setting up security assertions. The AH and ESP provides authentication and encryption, respectively. The IPSec protocol details are published in RFC's 2401 through 2412. IPSec is designed to work with both IPV4 and IPV6. In IPV4, IPSec is an add-on. IPV6 supports IPSec natively. Strictly speaking, IPSec is

mandatory for IPV6. IPSec works in two modes: Tunnel mode and Transport mode. Tunnel mode is typically used to tunnel traffic between two network gateways. Transport mode is used to protect traffic between two hosts.

IPSec stack is implemented natively in the kernel or as an add-on either as BITS (Bump In The Stack) or as BITW (Bump In The Wire), as illustrated in Fig. 4. If a kernel does not natively support IPSec, support can be added by introducing a software stack between the IP stack and the network device drivers. This mechanism is termed BITS. If the implementation is done using an external piece of hardware, it is called BITW. For example, the openswan project [18, 19] developed software for IPSec to be used as BITS.

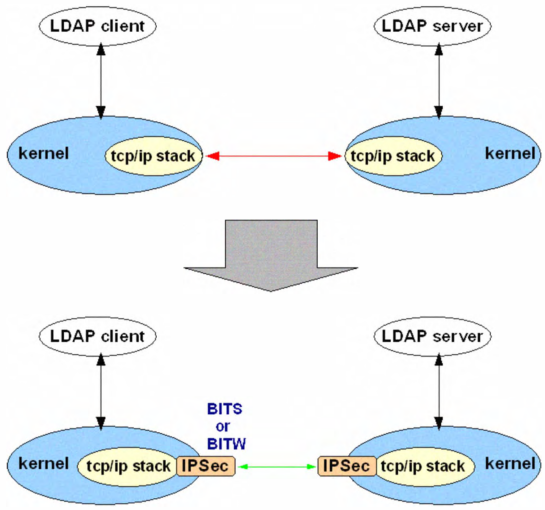

Figure 4: IPSec details for an LDAP connection

The IPSec function can be split into two main categories: *packet handling* and *trust relationship management*. Packet handling is usually done by the kernel itself. The kernel deals with packets that comes in and goes out. Security policies dictate the management of IPSec connections that the kernel is aware of. Security policies are stored in the Security Policy Database. For each connection established, the kernel has to keep track of the values of various parameters. The set of parameter values for a single IPSec connection is called an IPSec Security Association (SA). The SA's are stored in a Security Association Database.

Another security aspect of IPSec is the creation of a trust relationship between hosts by the creation of a secure communication channel and exchange of cryptographic keys. The details of this constitute IKE as specified in RFC 2409 [20]. IKE is implemented as a process (called *userland daemon*) that runs continuously and listens for IPSec connection requests.

III. PERFORMANCE EVALUATIONS

In this research, we implemented the three security protocols (LDAPS, SSH and IPSec) in a real network test-bed. Furthermore, we evaluated and compared the performance of these protocols. In this section, we present our experiment setup and results.

A. Experiment Setup

The experiment test-bed is shown in Fig 5. This is a self-contained environment, so it is not affected by other network traffics. All the systems are connected using IP addresses in the same subnet. The IP prefix is 172.16.x.x. The OpenLDAP [21] software was installed on the LDAP server machine. PERL with Net::LDAP [22] module was used on the client machine. The monitoring system was installed with Wireshark network analyzer [1] software and was used to monitor network traffic between the LDAP client and server.

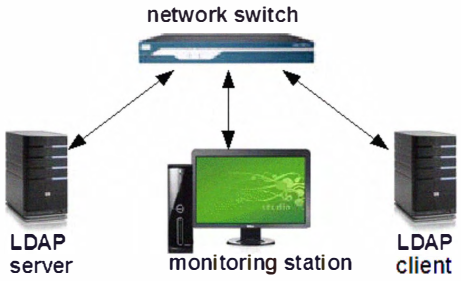

Figure 5: The test-bed setup

The details of the equipments are as follows:

LDAP Server machine
Intel-based server
CPU: 2 CPU's 3.2GHz (each), Memory: 2 GB
OS: Centos 5
NIC – Gig Ethernet interface
Static IP: 172.16.16.15

LDAP Client machine
Intel-based server
CPU: 2 CPU's 3.2GHz (each), Memory: 2 GB
OS: Centos 5
NIC – Gig Ethernet interface
Static IP: 172.16.16.16

Monitoring station
Intel-based workstation
CPU: 1.7GHz, Memory: 512 MB
OS: Windows XP
Static IP: 172.16.16.17
Software: Wireshark network analyzer

B. *Experiment Details and Results*

In the experiments, we mainly studied the delay performance of the security protocols. That is, we compared the end-to-end delay of the protocols. We run the experiments for different sizes of the directory, including 1, 10, 20, 50, 100, 200, 500, 750 and 1000 records. For each configuration (directory size + security protocol), the delay was measured for 10 runs and the average result is used.

First, clear text LDAP timing was used to establish the baseline for delay performance. A PERL program was used to bind to the LDAP server and search all records in the directory. The second set of experiments use LDAPS, and the LDAP server had to be configured with SSL. For this, OpenSSL [23] tools were used. The following tasks were done for LDAPS experiments:

1. A local Certificate Authority (CA) was set up and root certificates were generated for the local CA. (This was done so that server certificates could be signed locally instead of using a commercial CA.)
2. Server certificates were generated.
3. Server certificates were signed using root certificates of the local CA.
4. OpenLDAP server was configured to trust the local CA.
5. The openLDAP server configuration files were updated to specify the location of the certificate files.
6. OpenLDAP server was started to run on port 636 (using SSL) and on port 389 without encryption. This had to be done as root (administrative account) because ports up to 1024 are available only to administrator accounts.

The details of these configurations can be found in [24] and [25]. After these steps, the PERL program was modified to use LDAPS.

The third set of experiments was based on SSH. Port forwarding was set from the LDAP client on port 389 to port 389 on the LDAP server. The following command was used:

```
ssh -2 –f -NL 389:localhost:389 172.16.16.15
```

This command also required the use of root (administrative) account since the port number 389 is less than 1024. Once the port forwarding was set up, the PERL program was run using the plain text version.

Finally, IPSec was configured on both LDAP server and LDAP client. IPSec was used in transport (host-to-host) mode. CentOS kernel 2.6 comes with NETKEY/XFRM IPsec stack. Since the stack supports the XFRM interface for key management, any userland software that supports the protocol could be used. In this project, the 'racoon' daemon (from KAME project [26]), which implements the IKE protocol of IPSec, was used for key management.

We list the comparison of network stacks and security algorithms used by the four protocols in Tables I and II respectively. In order to use LDAPS, we slightly modified an LDAPS API (the function call for ldap connection had to be modified with proper parameters for ldaps). Proper keys had to be generated and configured for the LDAP server side. In the case of SSH, a user account on the server system is needed to make the connection between the client and server systems. This user account is in fact used to log into the server system for establishing the SSH tunnel. IPSec does not need any change to the software. However, setting up the IPSec transport between the two systems requires administrative privileges on both systems.

TABLE I. COMPARISON OF NETWORK PROTOCOLS

	Network Protocol	*Transport Protocol*	*Port*
LDAP	IP	TCP	389
LDAPS	IP	TCP	636
SSH	IP	TCP	22
IPSec	IP	None	n/a

TABLE II. COMPARISON OF SECURITY ALGORITHMS

	Authentication	*Encryption*	*Hash*
LDAP	None	None	None
LDAPS*	RSA	AES256	SHA1
SSH	RSA	AES128	SHA1
IPSec	Secure Hash	3 DES	SHA1
* : Net::LDAP uses the same ciphers as OpenSSL			

TABLE III. COMPARISON OF DELAYS

# of records	LDAP	LDAPS	SSH AES128	SSH AES256	IPSec 3DES	IPSec AES256
1	0.178	0.222	0.180	0.181	0.180	0.180
10	0.194	0.239	0.196	0.197	0.195	0.195
20	0.210	0.257	0.212	0.213	0.211	0.212
50	0.266	0.320	0.273	0.276	0.271	0.270
100	0.377	0.442	0.394	0.398	0.387	0.390
200	0.597	0.702	0.645	0.647	0.640	0.642
500	1.236	1.454	1.367	1.369	1.353	1.364
750	1.777	2.075	1.973	1.980	1.954	1.962
1000	2.311	2.704	2.585	2.594	2.566	2.573

The end-to-end delay results of the four sets of experiments are presented in Table III, where the unit is second. For illustration purpose, we also plot the same results for record 1 - 100 in Figure 6. From Table II we can see that the default encryption algorithms used by the three protocols are different. In order to find out how much the encryption algorithms contribute to the delay, we also measured the delay when SSH and IPSec use AES256. The results of using AES256 are also reported in Table III and plotted in Figure 7. From the experiments results, we obtain several observations:

1) IPSec is the fastest security protocol and it always has less delays than both LDAPS and SSH. This is mainly

because IPSec does not use public-key (RSA) authentication, which takes some time.

2) The delays of IPSec and SSH are very close to the clear-text protocol LDAP. That is, IPSec and SSH have very small security overhead in terms of delay.

3) Using different encryption algorithms only has very small effects on the end-to-end delay. For example, using AES256 only incurs a little longer delay than AES128 or 3DES.

4) As the number of records increases, the differences of *delay-per-record* diminish among the protocols. This is because the security overhead (encryption & authentication) is amortized over a large number of records.

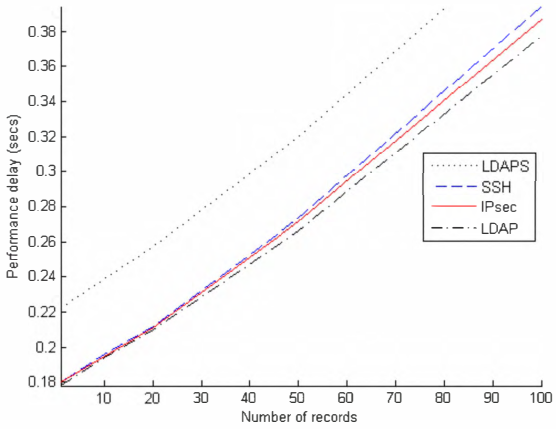

Figure 6: Delay comparison with default encryption algorithms

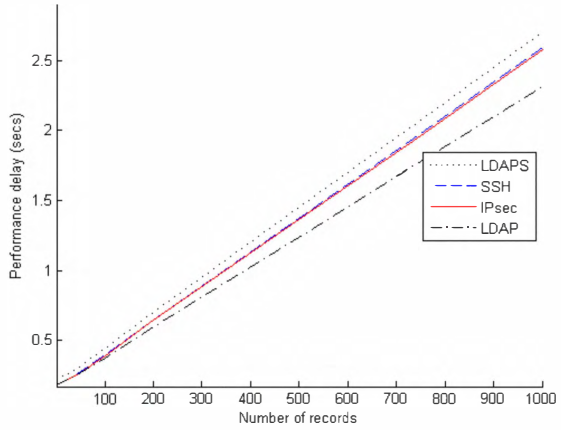

Figure 7: Delay comparison when using AES256

IV. CONCLUSIONS

This is paper, we studied the performance of several popular security protocols used in multi-tiered enterprise web applications. Specifically, we compared the end-to-end delays of deploying LDAPS, SSH, and IPSec by real-network experiments. Our experiments revealed several important issues: IPSec and SSH have very small security overhead and they are faster than LDAPS; the choice of different encryption algorithms has small effects on the end-to-end delay; the overhead of the security protocols is amortized when the number of records is large. Our results provide useful guidelines for the design and deployment of large scale multi-tiered enterprise web applications.

V. ACKNOWLEDGEMENT

This research was supported in part by the US National Science Foundation (NSF) under grants CNS-0963578, CNS-1002974 and CNS-1022552, and the US Army Research Office under grant W911NF-08-1-0334.

REFERENCES

[1] Wireshark network protocol analyzer. http://www.wireshark.org

[2] Rubin, D, A., and Geer Jr., E,D., *A Survey of Web Security*, Computer, Vol 31, Issue 9, Sep 1998, pp 34-41

[3] McDaniel, P., and Rubin, D.A., *Web Security (editorial),* Computer Networks, Vol 48, Issue 5, August 2005, pp 697-699

[4] Kruegel, C., Vigna, G., and Robertson, W., *A multi-model approach to the detection of web-based attacks,* Computer Networks, Vol 48, Issue 9, Augut 2005, pp 717-738

[5] Potter, B., *High Time for Trusted Computing,* IEEE Security and Privacy, Vol. 9 No. 6, November/December 2009, pp. 54-56

[6] RFC 2818 – HTTP over TLS. http://www.ietf.org/rfc/rfc2818.txt

[7] RFC 1777 - LDAP – Lightweight Directory Access Protocol. Available at http://www.ietf.org/rfc/rfc1777.txt

[8] RFC 1823 - LDAP Application Program Interface. http://www.ietf.org/rfc/rfc1823.txt

[9] Barrett. D., Silverma, R, and Byrnes, R., *SSH, The Secure Shell: The Definitive Guide (2nd edition),* O'Reilly Media Inc., 2005.

[10] Doraswamy, N., and Harkins, D., *IPSec (2nd edition)*, Prentice Hall, 2003.

[11] RFC 1487 – First protocol definition of LDAP. http://www.ietf.org/rfc/rfc1487.txt

[12] RFC 2798 – Defnition of the inetOrgPerson LDAP object class. Available at http://www.ietf.org/rfc/rfc2798.txt

[13] CAS – Central Authentication Service. http://www.jasig.org/cas

[14] Shibboleth – Federated Single Signon Software. http://shibboleth.internet2.edu

[15] Openssh software. Available at http://openssh.org

[16] Well known ports 0-1023. http://www.iana.org/assignments/port-numbers

[17] Kolesnikov, O. and Hatch, B., *Building Linux Virtual Private Networks*, pp. 30, New Riders, Indiana, 2002.

[18] Wouters, P. and Bentoft, K., *Building and Integrating Virtual Private Networks with Openswan*, PACKT Publishing, 2006.

[19] Openswan home page. http://openswan.org

[20] RFC 2409 – The Internet Key Exchange. http://www.ietf.org/rfc/rfc2409.txt

[21] OpenLDAP. Available at http://www.openldap.org

[22] Net::LDAP module. http://search.cpan.org/~gbarr/perl-ldap-0.39/lib/Net/LDAP.pod

[23] OpenSSL. http://openssl.org

[24] OpenLDAP SSL Configuration – Part 1 blogs.sun.com/hariblog/entry/openldap_ssl_configuration_part_1

[25] OpenLDAP SSL Configuration – Part 2 blogs.sun.com/hariblog/entry/openldap_ssl_configuration_part_2

[26] KAME Project home page. http://www.kame.net