

MGT 6335 Group Presentation

Securing HTTP
Securing SMTP using TLS
Securing FTP using SSH

Ryan Moore
Joseph Kahlich
Mike Gordon

Securing Web Site Network Traffic

- *Hyper Text Transfer Protocol (http) is the standard protocol for web browsers*
- *Web pages are written in HTML and HTTP is used to transfer the pages from the webserver to the client (webbrowser)*
- *This protocol is seen at the beginning of all web address in the browser.*
- *<http://www.udallas.edu> for example.*
- *In order for Ecommerce to be viable on the internet it must be secure*
- *The presentation will discuss 2 ways of securing HTTP transactions over the internet*
 - Secure Socket Layer (SSL)
 - Secure HTTP

Secure Socket Layer

- **SSL was invented by the Netscape corporation.**
- **Originally designed to secure the Application Layer protocols of HTTP, LDAP, and POP3**
- **SSL addresses the 3 main concerns of security: Confidentiality, Integrity, and Authentication**
- **Confidentiality- Data transferred using SSL is encrypted by symmetric key algorithms.**
- **Integrity - Secure Hash Functions insure that data cannot be tampered with during an SSL session.**
- **Authentication – Public Key cryptography allows SSL clients and servers to authenticate each other.**
- **SSL 3.0 was released by Netscape in 1996.**
- **TLS 1.0 (Transport Layer Security) is the standards based successor to SSL and is discussed later in the presentation**

SSL Protocols

- **SSL is actually a collection of protocols that work together to provide end to end secure communications.**
- **SSL Record Protocol**
 - Transports the other SSL protocols and to transfer application data within the SSL session.
- **SSL Alert Protocol**
 - Transfers messages about errors in communication.
 - These errors can be flagged as Warning or Fatal.
 - A Fatal Error will end the SSL Session
- **Change Cipher Spec protocol**
 - Simplest of all the SSL protocols
 - Causes a Pending Session State to become the fixed session state
 - This occurs when the client and server have agreed on a set of cryptographic protocols
- **SSL Handshake Protocol**
 - Most complex SSL protocol
 - Starts the session between the client and server
 - Negotiates what encryption parameters will be used for the session

SSL Transaction

- **Initiating an SSL Session is broken up into 4 phases.**
- **Phase 1**
 - Client contacts the server
 - Informs the server of the cryptographic parameters it supports
- **Phase 2**
 - The server sends its digital certificate to the client for verification
 - The server may also request the clients certificate, but this is optional.
- **Phase 3**
 - The client verifies the servers Digital Certificate with the Certificate Authority that issued it.
 - Client and Server then begin the Key Exchange process for the selected Symmetric Key Algorithm
- **Phase 4**
 - The client and server exchange messages using the agreed upon cryptographic method.
 - If each can read the others message then secure communication has been established
- **The initiation is complete and application data can then be securely transferred.**

S-HTTP

- **Secure HTTP (S-HTTP) was developed in 1994 by Enterprise Integration Technology**
- **It is defined in RFC 2660**
- **S-HTTP is an extension of the HTTP protocol.**
- **An S-HTTP aware client and server are necessary for secure transactions**
- **A S-HTTP aware client can communicate with non S-HTTP server and vice versa.**
- **Has different Design goals than SSL**
 - S-HTTP is designed to send secure individual messages between the client and server
 - SSL is designed to establish a secure connection between the client and server.
 - S-HTTP encrypts messages at the Application Layer, while SSL encrypts at the transport layer.

S-HTTP Transactions

- *S-HTTP messages consist of three parts*
 - The actual HTTP message
 - The Senders Cryptographic Parameters
 - The Receivers Cryptographic Parameters
- *Sender and Receiver work out a common set of cryptographic Parameters to use for each message.*
- *S-HTTP fully supports Public Key Infrastructure but does not require it.*
- *S-HTTP defines 2 key exchange methods*
 - Public Key enveloped key exchange
 - Pre-defined symmetric keys

SSL and S-HTTP

- ***SSL and S-HTTP are fully compatible***
- ***SSL can be used to in-conjunction with S-HTTP as they function at different layers***
- ***S-HTTP has been perceived as “too flexible”***
- ***Allowing implementations that are inherently insecure.***
- ***Few Major Web Browsers and Servers implement S-HTTP***
- ***SSL has become the defacto Internet Standard for securing HTTP Transactions***

SMTP and TLS for Secure E-mail

- ***SMTP (RFC 2821) Simple Mail Transfer Protocol.***
 - Objective: To transfer mail reliably and efficiently.
 - Problem: Messages are passed in clear text.
 - Possible solution: Transport Layer Security.
- ***TLS (RFC 2246) Transport Layer Security.***
 - Objective: Provide privacy and data integrity between two communicating applications.
 - 2 Layers to the protocol: Record and Handshake.
 - 2 Properties of the Record layer:
 - Privacy using encryption such as DES and RC4.
 - Integrity using MAC's such as SHA and MD5

SMTP and TLS

- **Secure SMTP over TLS (RFC 2487)**
- **Objective: Provide private and secure SMTP communication across the Internet.**
- **STARTTLS: Extension of SMTP that is similar to HTTPS for HTTP traffic as mentioned earlier in the presentation. Considered to be the “verb” of secure SMTP. STARTTLS is also the name of the SMTP secure service and the command which initiates a secure connection between entities.**
- **Paul Hoffman: Author of RFC 2487. Cordially accepted my phone call. He described STARTTLS as similar to HTTPS. An initial secure connection attempt is made, if it fails the entities may either fall back to SMTP or terminate the connection.**
- **STARTTLS may be initiated between client and server or server and server.**
- **STARTTLS does not require certificates by using Anonymous Diffie-Hellman**

SMTP and TLS

- ***Possible Insecurities:***

- STARTTLS does not authenticate the originator of the email.
- A man-in-the-middle attack can be launched by deleting the "250 STARTTLS" response from the server. This would cause the client not to try to start a TLS session.
- It should be noted that SMTP is not an end-to-end mechanism. Thus, if an SMTP client/server pair decide to add TLS privacy, they are not securing the transport from the originating mail user agent to the recipient.

SMTP and TLS

- *Log sample of a mail server with STARTTLS enabled. Verify=FAIL is fine since certificates are not required.*

Jul 10 17:12:39 mail sm-mta[52532]: STARTTLS=server, relay=athos.abc.xyz.edu [122.16.15.40] (may be forged), version=TLSv1/SSLv3, verify=FAIL, cipher=EDH-RSA-DES-CBC3-SHA, bits=168/168

Securing FTP

- **Two primary methods**

- Replacing traditional FTP with HTTPS file uploads via a web browser.

- Does not actually use the FTP protocol but instead uses HTTPS as previously described

- Utilizing the Secure Shell (SSH) protocol

- Uses a client server SSH application to provide an encrypted tunnel between the client and server
- SSH uses hashing and key exchanges to provide authentication between the devices
- SSH has configuration risks that could bypass firewall controls

The remaining charts will focus on SSH, how it works and how it should be configured

What is SSH

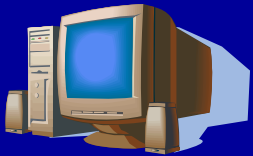
- **Software based approach to network security**
- **Transparently encrypts data in transit across the network between systems**
- **SSH is a protocol, however there are implementations of the protocol in products also named SSH.**
- **The protocol provides the following:**
 - Secure sessions utilizing secure encryption
 - Full, secure replacement for FTP, Telnet and some UNIX commands
 - Multiple high security algorithms and strong authentication methods that prevent security risks such as spoofing and eavesdropping
 - Multiple ciphers including 3DES, AES, and Blowfish
 - Transparent tunneling of X.11
 - Automatic and secure authentication of both client and server
- **The current version of SSH is SSH version 2 or SSH-2**
 - SSH-2 protocol is in draft in the IETF SECSH working group
- **SSH version 1 (SSH-1) has several known weaknesses and vulnerabilities and should not be avoided**
 - For example, SSH-1, uses a 32-bit cyclic redundancy check (CRC-32) for integrity checking as opposed to MD-5 or SHA-1

SSH Features

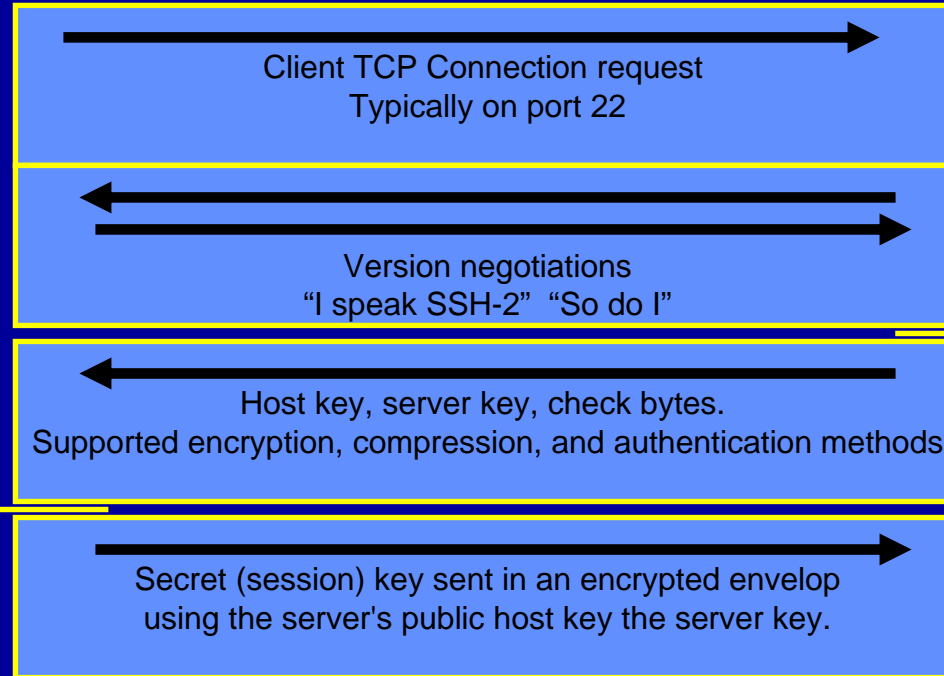
- **Privacy**
 - End-to-end encryption based on random keys issued per session
 - Encryption: AES, ARCFOUR, Blowfish, DES, IDEA, 3DES.
- **Integrity of communications**
 - Cryptographic integrity checking verifies data hasn't been altered
 - Uses hash algorithms based on MD5 and SHA-1
- **Authentication**
 - User and Server authentication
- **Authorization**
 - Possible to restrict client access
- **Forwarding**
 - Encapsulating other TCP-based service, such as FTP within an SSH session

Establishing an SSH Session

CLIENT



SERVER



Both sides compute a common 128-bit *session identifier*. MD5 hash of the host key, server key, and check bytes

Client waits for server authentication. Then Encryption Starts

Switch to a packet-based protocol over the underlying TCP connection. 32-bit length field, 1-8 bytes of random padding, a one-byte packet type code, the packet payload data, and a four-byte integrity check field.

Note: SSH-2 key exchange only permits diffie-hellman-group1-sha1 for server authentication.

SSH Forwarding: A Security Risk

– Port Forwarding

- Local and remote
- Forwards traffic from one port to another

– Agent Forwarding

- Allows connections from one computer, through a second computer, and onto a third using public-key authentication, but without installing a private key on the second machine
- Can lead to SSH reverse tunnels to the trusted network with forwarding through an un-trusted network such as a DMZ

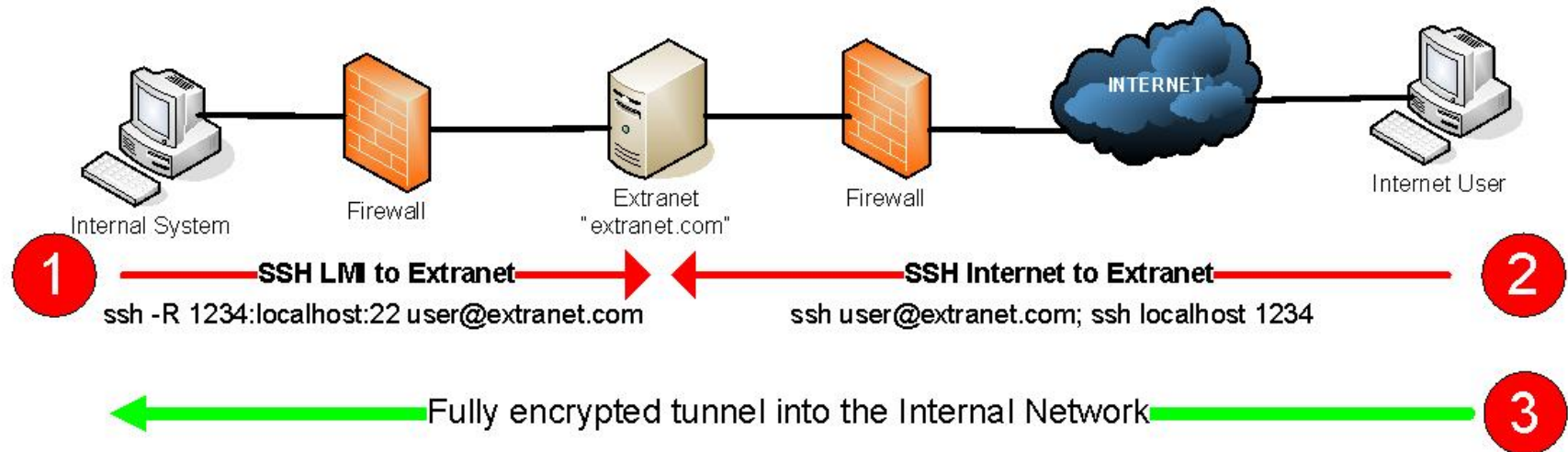
– X.11 Forwarding

- Returns remote GUIs

The use of forwarding bypasses firewall policies that may restrict connections

Problem: SSH Reverse Tunnel

SSH Reverse Tunnels: Example Problem Scenario



- Example Scenario: Simultaneous SSH connections from Internal Network to Extranet and from Internet to Extranet allows for direct inbound path.
- General: Simultaneous SSH connections inbound/outbound, Internal network to Extranet and inbound/outbound Extranet to Internet.
- Root Issue: Using SSH, it is possible to create fully encrypted tunnels INBOUND to the internal network.

Configurations to Protect

(*sshd_config* refers to SSH Daemon configuration file)

When Simultaneous connections are possible the SSH server should be configured such that:

1. **Extranet SSH server and configuration only under tight control**
2. **Two-factor authentication into Extranet highly recommended**
3. **Allow only a small and finite set of IP addresses to connect from the Internet to the SSH service on Extranet Servers** (implement at extranet firewall).
4. **Only use SSH Protocol Version 2** (implemented on SSH server)
sshd_config: Protocol 2
5. **Disable TCP Forwarding** (implemented on SSH server)
sshd_config: AllowTcpForwarding no
6. **Disable AgentForwarding** (implemented on SSH server)
sshd_config: AgentForwarding no
7. **Do not allow direct root login** (implemented on SSH server)
sshd_config: PermitRootLogin no
8. **After configuration is set, set permissions of sshd.conf to 400 (read-only for root/administrator)**

References

- **SSH, The Secure Shell: The Definitive Guide (Barrett & Silverman, O'Reilly Press)**
- **Building Internet Firewalls (Chapman & Zwicky, O'Reilly Press)**
- **Wikipedia Entry on SSL,
http://en.wikipedia.org/wiki/Secure_Sockets_Layer**
- **Schiffman, A. (1999) RFC 2260 - The Secure Hypertext Transfer Protocol, <http://www.apps.ietf.org/rfc/rfc2660.html>**
- **Kangus, Erik (2005) How does SSL Work?,
http://luxsci.com/info/about_ssl.html**
- **Allen, c. Dierks, T.(1999)RFC 2246 - The TLS Protocol Version 1.0.
<http://www.faqs.org/rfcs/rfc2246.html>**
- **Klensin, J.(2001)RFC 2821 - <http://www.faqs.org/rfcs/rfc2821.html>**
- **Hoffman P.(1999)RFC 2487 - SMTP Service Extension for Secure SMTP over TLS <http://www.faqs.org/rfcs/rfc2487.html>**
- **Hoffman P.(2005)Phone interview by Joseph Kahlich**
- **<http://www.ssh.com>**