

Lecture 15: Man in the Middle Attack to get Passwords from HTTPS Sessions

How HTTPS Works

HTTP v. HTTPS

HTTP doesn't encrypt data at all

1. You can sniff traffic with Wireshark, ettercap, etc.
2. Completely insecure

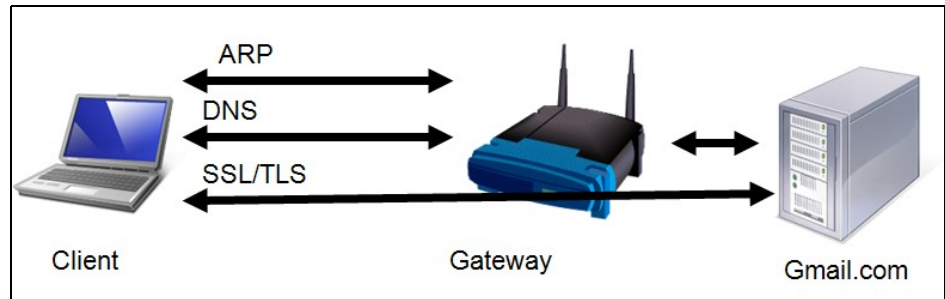
HTTPS uses public-key encryption to secure data

3. Much safer, but it can still be cracked to some extent by a man-in-the-middle attack

Components of HTTPS

When you use a secure session (HTTPS), these protocols work together:

4. Address Resolution Protocol (ARP)
5. Domain Name System (DNS)
6. Secure Sockets Layers (SSL)



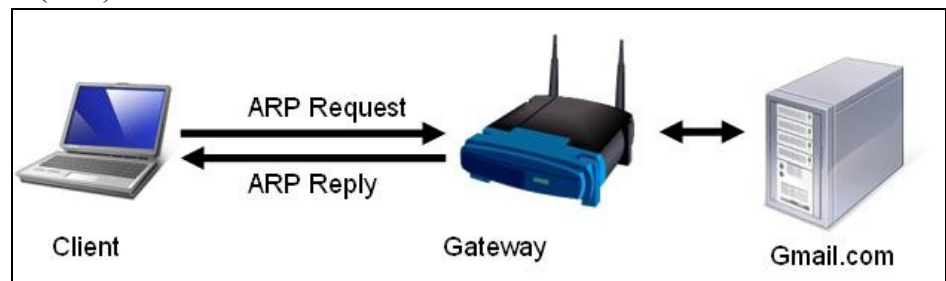
ARP Request and Reply

Client wants to find Gateway

ARP Request: Who has 192.168.2.1?

ARP Reply:

7. MAC: 00-30-bd-02-ed-7b has 192.168.2.1



Demonstration: Sniffing ARP with Wireshark

Start Wireshark capturing packets

Clear the ARP cache

8. `arp -d *`

Ping the default gateway

Source	Destination	Protocol	Info
supermic_82:11:bc	Broadcast	ARP	who has 192.168.2.1? Tell 192.168.2.28
BelkinCo_02:ed:7b	supermic_82:11:bc	ARP	192.168.2.1 is at 00:30:bd:02:ed:7b

DNS Query and Response

Client wants to find

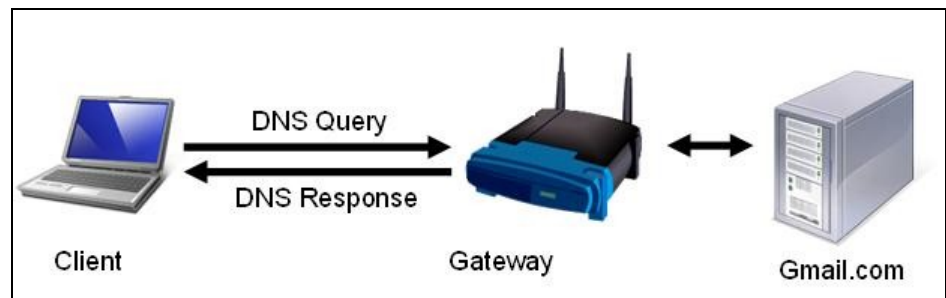
Gmail.com

DNS Query: Where is

Gmail.com?

DNS Response:

9. Gmail.com is at 64.233.171.83



Demonstration: Sniffing DNS with Wireshark

Start Wireshark capturing packets

Clear the DNS cache

10. `ipconfig /flushdns`

Ping Gmail.com

Lecture 15: Man in the Middle Attack to get Passwords from HTTPS Sessions

Source	Destination	Protocol	Info
192.168.2.28	192.168.2.1	DNS	Standard query A gmail.com
192.168.2.1	192.168.2.28	DNS	Standard query response A 64.233.171.83

SSL Handshake

SSL handshake has three stages:

11. Hellos
12. Certificate, Key Exchange, and Authentication
13. "Change cipher spec" – handshake finished



The Gateway just forwards all this traffic to the Web server

Demonstration: Sniffing SSL Handshake with Wireshark

Start Wireshark capturing packets

Open a browser and go to yahoo.com

Click the My Mail button

	Source	Destination	Protocol	Info
Hand	192.168.2.28	209.73.168.74	TCP	1180 > https [SYN] Seq=0 Len=0 MSS=1460
	209.73.168.74	192.168.2.28	TCP	https > 1180 [SYN, ACK] Seq=0 Ack=1 win=65535 Len=0
	192.168.2.28	209.73.168.74	TCP	1180 > https [ACK] seq=1 Ack=1 win=17520 [TCP CH
Hello	192.168.2.28	209.73.168.74	SSLv2	Client Hello
	209.73.168.74	192.168.2.28	TLSv1	Server Hello, Certificate, Server Hello Done
Key	192.168.2.28	209.73.168.74	TLSv1	Client Key Exchange, Change Cipher Spec, Encrypt
	209.73.168.74	192.168.2.28	TLSv1	Change Cipher Spec, Encrypted Handshake Message
	192.168.2.28	209.73.168.74	TLSv1	Application Data

Hand – these three packets are the TCP Handshake, which happens before the SSL handshake

Hello – these two packets are the Hellos, which start the SSL handshake

Key – these packets perform the last two stages of the SSL handshake:

14. Certificate, Key Exchange, and Authentication
15. "Change cipher spec" – handshake finished

Open a Socket to Port 443

This is the usual SYN, SYN/ACK, SYN TCP handshake

Port 443 is used for HTTPS

Hellos

Client Hello

Server sends Hello

16. This exchange is used to agree on a protocol version and encryption method

Certificate, Key Exchange, and Authentication

Server sends Certificate

Client sends Public Key

Client Authenticates Certificate with Certificate Authority (not visible)

Change Cipher Spec

Server sends "Change Cipher Spec"

Client sends "Change Cipher Spec"

SSL Handshake is done, now client can send encrypted Application Data

