Topics in Network Security

Jem Berkes
MASc. ECE, University of Waterloo
B.Sc. ECE, University of Manitoba

www.berkes.ca

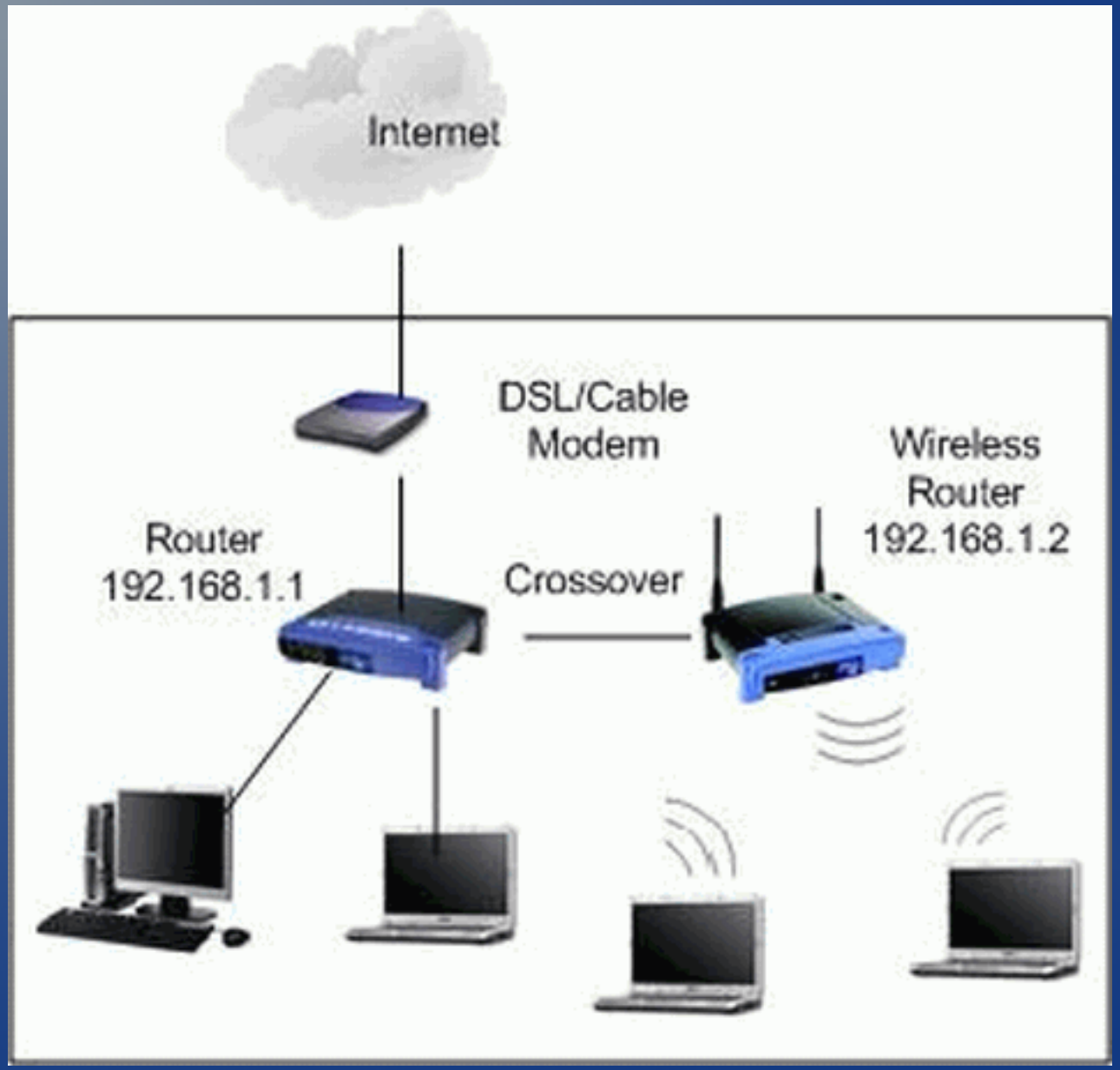
February, 2009

Ver. 2

In this presentation

- Wi-Fi security (802.11)
- Protecting insecure channels
- Case study: an insecure application
- How is a computer hacked?

Wi-Fi Security (802.11)



… Wi-Fi Security (802.11) ...

- As with other situations, two attack categories
  - PASSIVE: silently listening and reading signals
  - ACTIVE: modifying signals or affecting system
- Some threats are more specific to wireless
  - Radio jamming and interference
  - Unauthorized access or authentication

… Wi-Fi Security (802.11) ...

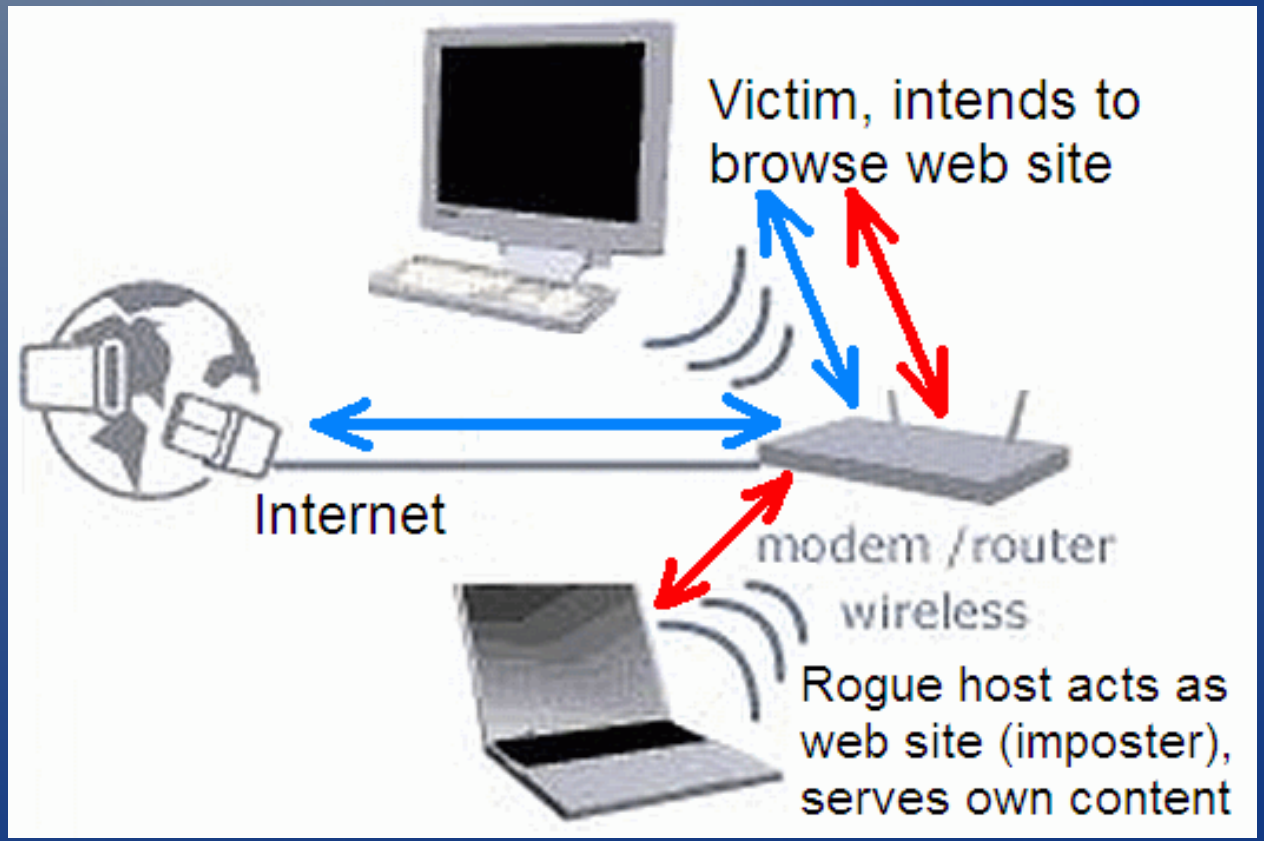
Passive eavesdropping

- Signals sent through air on public frequencies
- Eavesdrop using any wireless card!

… Wi-Fi Security (802.11) ...

Active attacks

- Many possible scenarios
- e.g. attacker places rogue host onto network



… Wi-Fi Security (802.11) ...

- Various 802.11 security standards
- WEP, WPA, WPA2...
- Your wireless LAN requires a password
  - Does that mean it's secure?

… Wi-Fi Security (802.11) ...

- Plenty of attacks are possible and practical
- WEP, Wired Equivalent Protocol (still around)
  - RC4 key easily discovered in about 1 minute
- WPA and WPA2, Wi-Fi Protected Access
  - Shared passwords under 13 chars breakable
    - Brute force speed rapidly improving
  - TKIP mode can be broken in a few minutes
    - No matter how strong the password
  - Might be safe: WPA and WPA2 using AES
    - Password must still be very strong

… Wi-Fi Security (802.11) ...

- Horrendous track record for Wi-Fi security
  - Latest critical attack published in Nov 2008
  - Largely protocol/math flaws, some brute force
- An average wireless LAN is likely insecure
  - Home or small office Wi-Fi likely exploitable
  - Configuring secure Wi-Fi is very challenging
  - Both my D-Link routers malfunction with WPA2
- We should not trust a wireless link for security
- Assume that Wi-Fi is an insecure channel

Protecting insecure channels

Wi-Fi is basically an insecure channel.

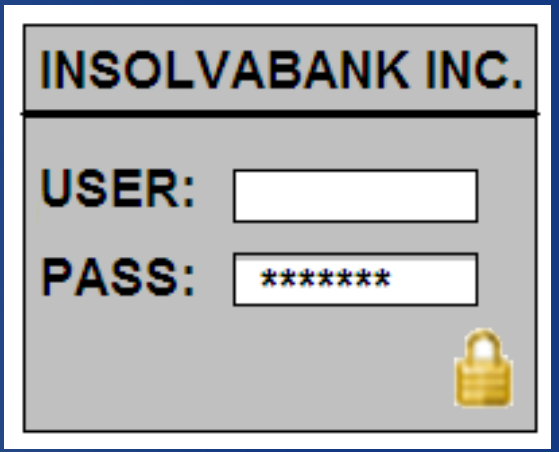
Ethernet packets on wired LAN can be sniffed too.

How do you protect data?

… Protecting insecure channels …

- Two elements to protecting IP traffic
  - Encryption (symmetric ciphers like RC4, AES)
  - Key exchange (RSA, DSA) and authentication
- Remember: encryption alone is not enough!
- Imagine a criminal sets up a web site that looks like your bank's, complete with SSL (lock icon)

… Protecting insecure channels …



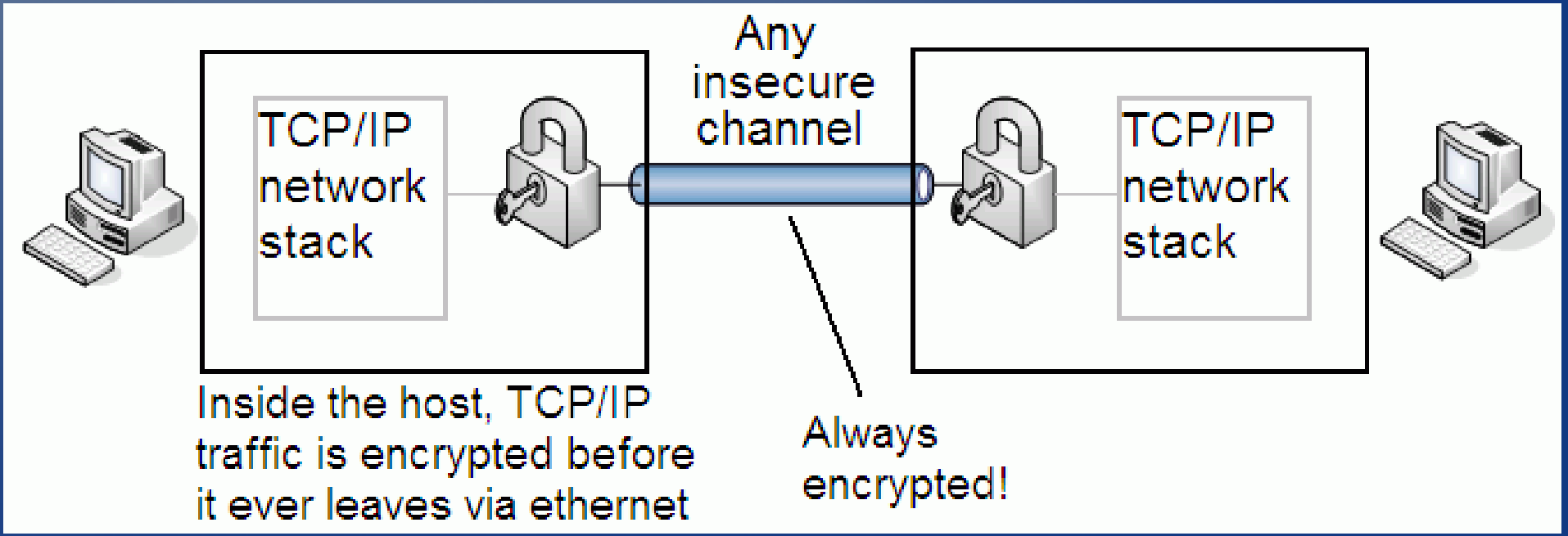
- Looks like your bank, looks secure
  - It's not hard to run SSL (https)
  - Encryption alone is not enough
  - They can still steal your password
- The ONLY thing that would alert you to fraud:
  - The address isn't your bank web site address
  - Or, there is a warning about the certificate
    - Certificate is invalid or doesn't match the domain
- Certificate authentication is essential !
  - Catches impostors, man-in-the-middle attacks

… Protecting insecure channels …

- Application layer solution
  - Transport Layer Security (TLS), previously SSL
  - Encrypts data so that it can not be sniffed
  - Also supports checking of certificates
    - Digital signature; authenticates identity
  - TLS is widely used in “https://” web sites
    - A cipher like 128-bit RC4 provides encryption
    - Site certificates provide authentication
    - Both must be used to achieve security!

… Protecting insecure channels …

- Tunneling solutions
  - IPsec, an OS-based tunnel for IP packets
  - Virtual Private Network (VPN) e.g. OpenVPN
  - Secure Shell (SSH) tunnel, easy to do



… Protecting insecure channels …

- Application-layer SSL/TLS is strong enough
  - The connection is safe even if the channel is not
- So why do you need tunnels at all?
  - Many applications fail to use SSL/TLS
  - Others make partial or incomplete use of it
    - e.g. Case study, coming up in presentation
  - Many https web sites fail to use total SSL/TLS
    - They often load images, content from plain http
    - Malicious attacks are still possible
  - When in doubt, safer to use tunnel for all traffic

… Protecting insecure channels …

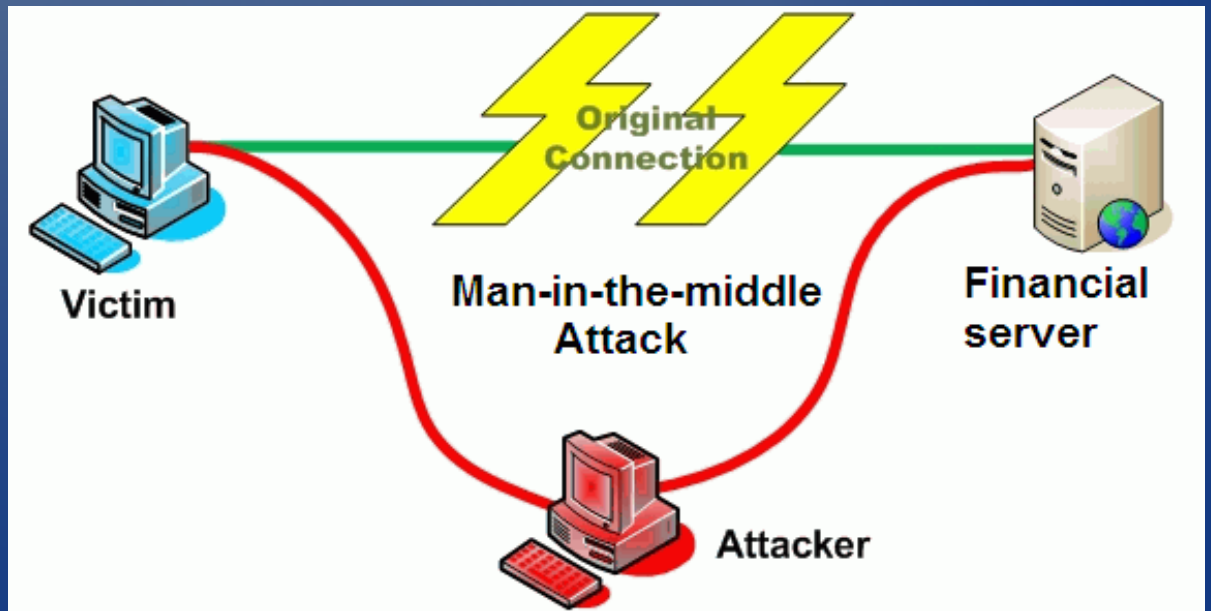
- SSH tunnel is easy using OpenSSH software
- ssh -L 1234:google.com:80 user@host
  - Opens ssh connection to host and logs in user
  - Forwards local port 1234 to google.com port 80
  - You can load http://127.0.0.1:1234 in browser
  - Your IP address does not connect to google
  - Instead, your traffic is encrypted over to host
  - The ssh host is the one contacting google.com

… Protecting insecure channels …

- ssh -D 1234 user@host
  - Open ssh (secure) connection to trusted host
  - Establishes a SOCKS proxy over ssh tunnel
  - In web browser, set proxy to 127.0.0.1:1234
  - All web traffic will be tunneled through host
  - That host opens new connections on demand
    - Your IP doesn't make TCP connections to sites
  - All traffic is encrypted before leaving your IP
  - Traffic leaving the ssh host can still be sniffed

Case study: an insecure application

- Real example: software from financial company
- Communicates very sensitive financial data
- Supposedly uses SSL, should be safe?
  - Turns out unencrypted data can still be sniffed
  - Failure to check certificates, so MITM possible

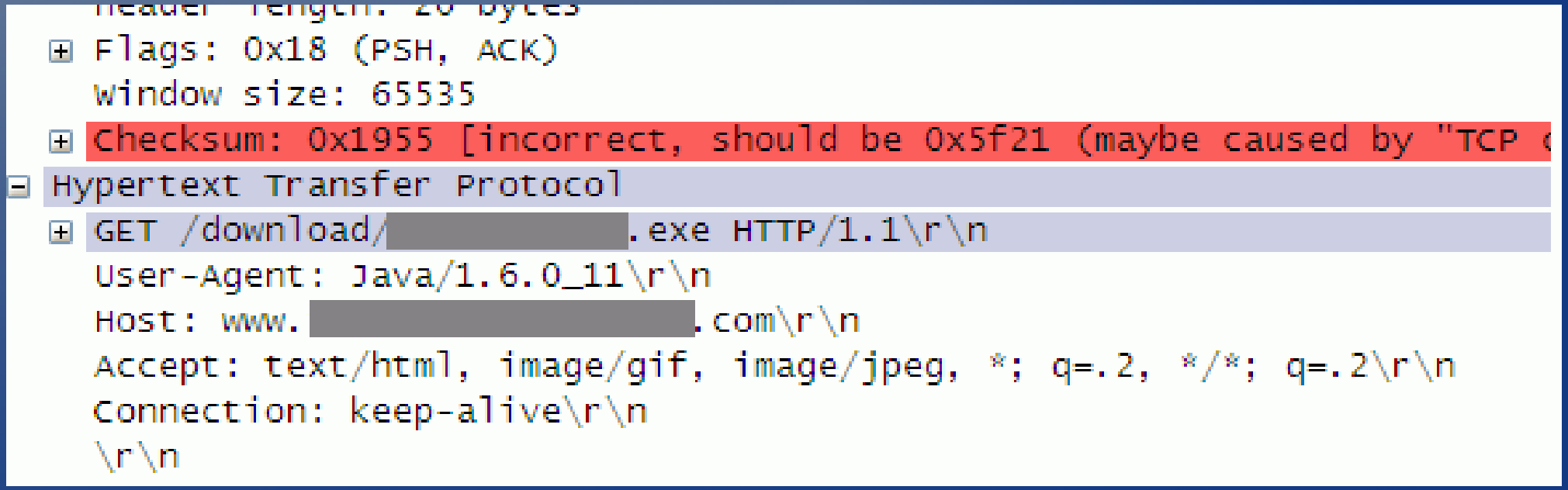


...Case study: insecure application...

- How to investigate?
- First step: capture packets
  - e.g. tcpdump on Linux, unix
  - Wireshark (used to be Ethereal)
- Capture ethernet traffic while doing “SSL login”

...Case study: insecure application...

- First thing I notice: some http connections
  - Application makes an http (not encrypted) connection to check for latest version. Wireshark decodes the http request.

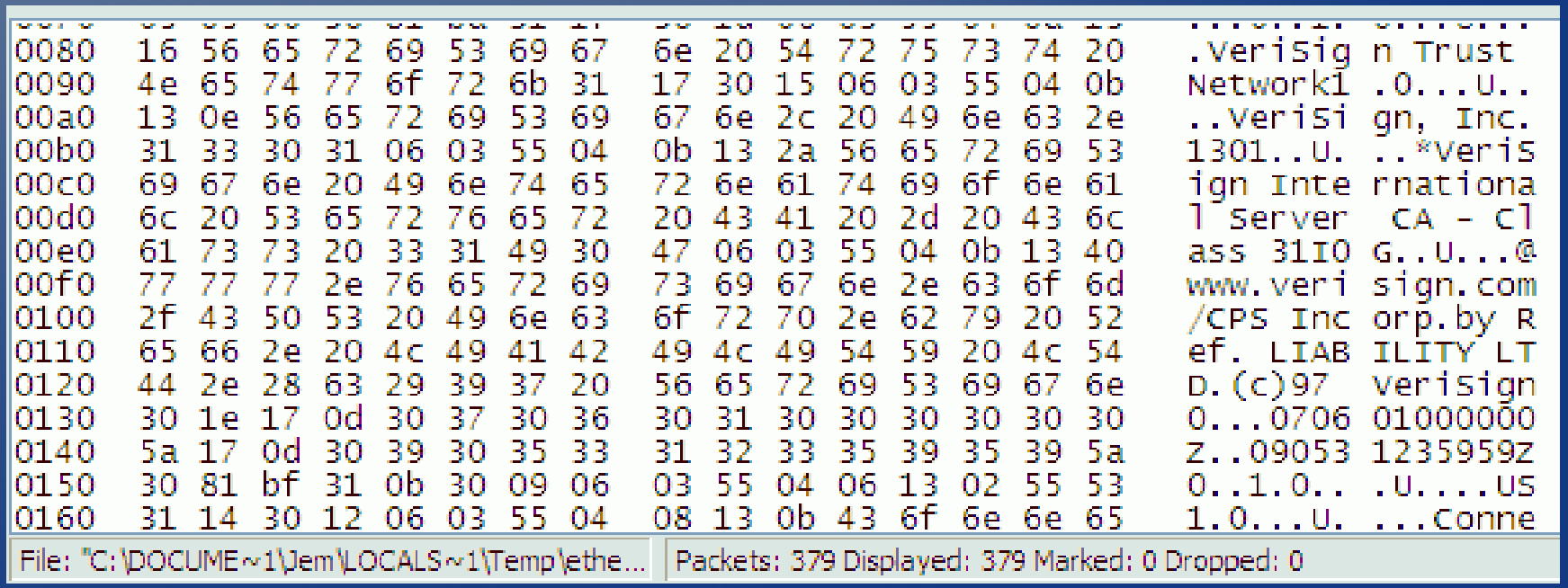


...Case study: insecure application...

- This request over the web is not encrypted, and neither is the reply (it is not SSL)
- Notice that this is a potential attack vector
  - An attacker could redirect this http to himself
  - Could interfere with application's mechanism to check its version and capabilities
  - Is this a threat? Very possibly.
- In any case, this connection should be over SSL/TLS. The software is in “SSL mode” !

...Case study: insecure application...

- Second connection seen: tcp port 8001 (means nothing), but cleartext ASCII data is visible
- The data being received from the server looks like a TLS certificate which is likely part of the negotiation at the start of SSL/TLS



...Case study: insecure application...

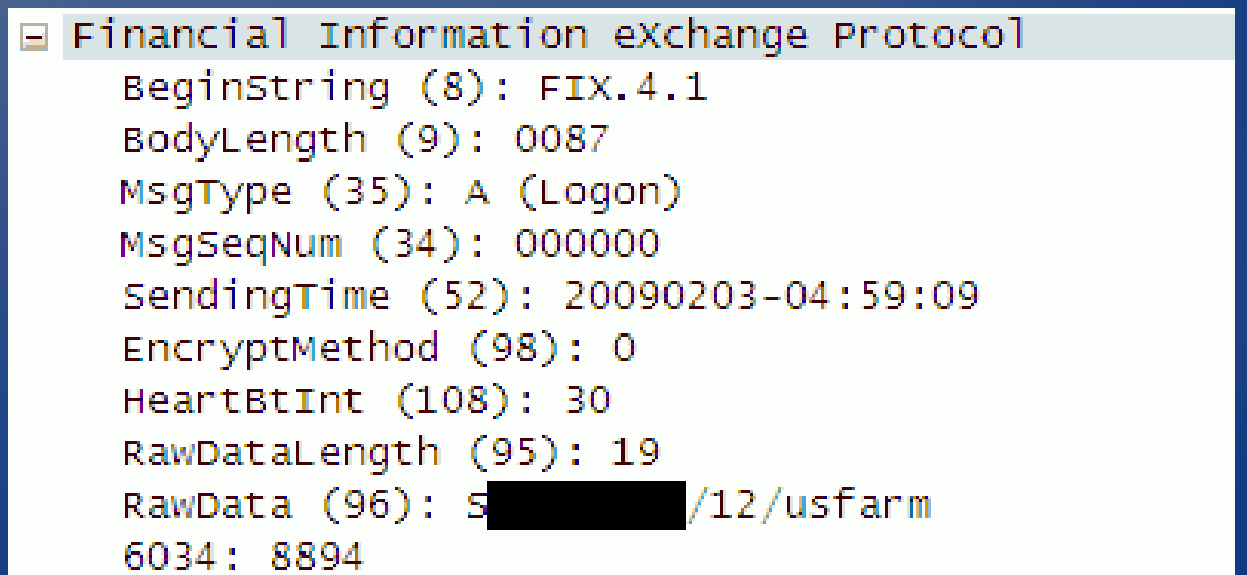
- In Wireshark, select only this port traffic by using display filter: tcp.port == 8001
- The rest of the packets all contain unreadable binary data (encrypted?). This is good news.
- It does appear that this port 8001 traffic is the SSL traffic which the application claims to use. This is an educated guess.

...Case study: insecure application...

- But there are further TCP/IP connections to inspect: port 8000. Again tell Wireshark to use display filter: tcp.port == 8000
- This is where things get ugly...
- Virtually all of these packets contain readable ASCII data. It is definitely not encrypted, and there is no sign of a certificate.
- Some of the visible (sniffable) data is financial in nature. It's not private, but it is definitely financial and definitely in the clear.

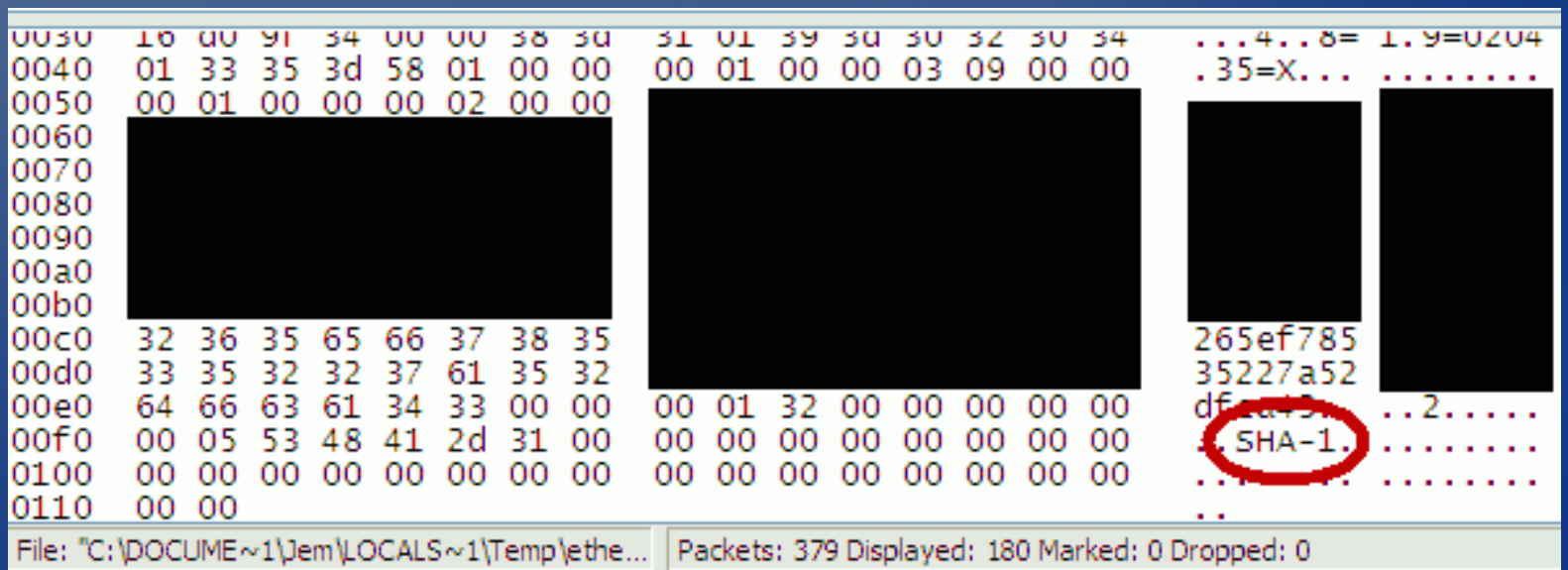
...Case study: insecure application...

- Wireshark even identifies it as “Financial Information eXchange Protocol” and a user name is readable!
- This user name is, in fact, transmitted many times in the clear... something that should never happen when we are expecting “SSL” mode!



...Case study: insecure application...

- One of the packets contains something truly interesting; user name (in the clear) combined with what looks like the hash of the password.
- The word SHA-1 appears; this is a hash algorithm and the hexadecimal ASCII format data dump looks a lot like a hash output.



...Case study: insecure application...

- We take an educated guess that the application is transmitting the hash of the password
- Transmitting the hash of a password is safer than sending the password in the clear; however, it can still be a bad idea.
- Depending on implementation, this kind of data could be abused by an attacker or even used to gain account privileges.

...Case study: insecure application...

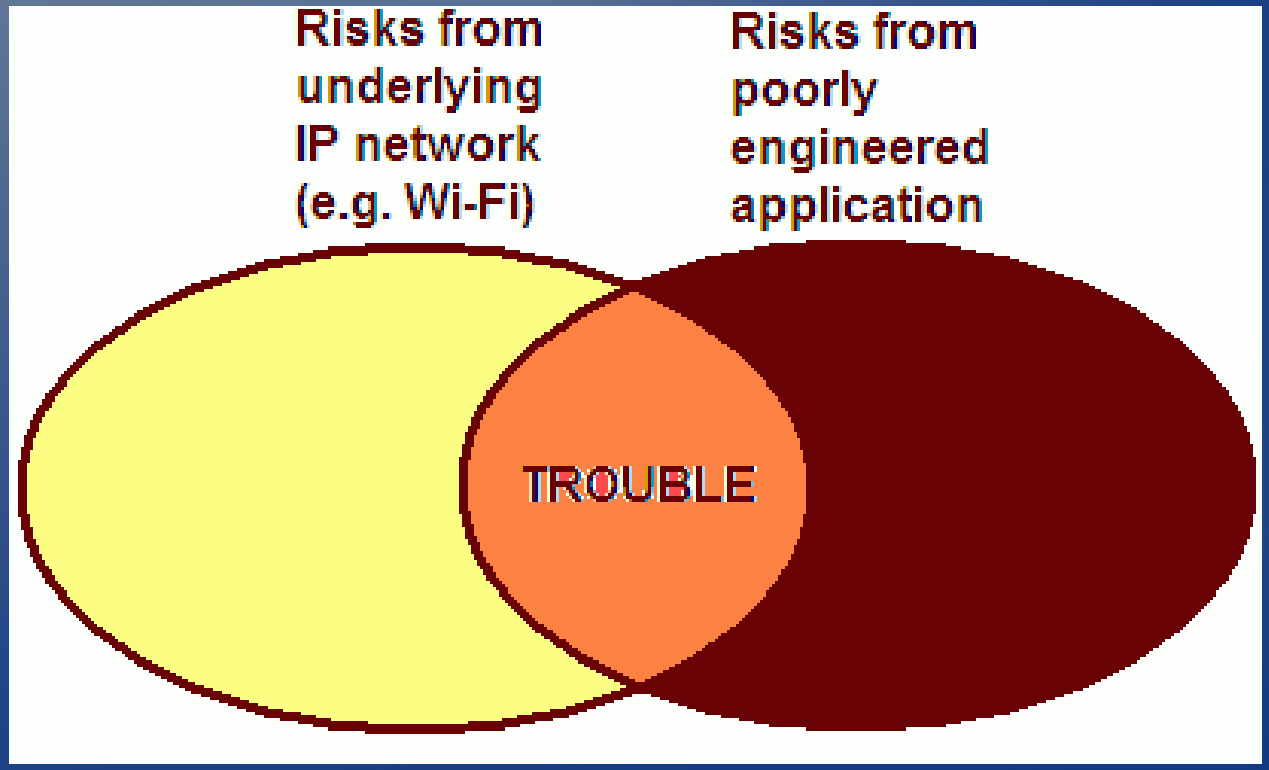
- What some simple packet dumps have showed:
  - While one connection is in fact SSL/TLS, other non-SSL connections are made too
  - Those unprotected connections contain sensitive data, including user names. The password may be compromised too.
  - All the unencrypted connections have no certificate and could be spoofed, or attacked by a man-in-the-middle (MITM)
  - The software is misleading people if they presume it is SSL enabled and secure.

...Case study: insecure application...

- Keep in mind, this particular software is used by many people from a major financial company.
- What we can learn from this case study:
  - Even “SSL-enabled” software can make poor use of SSL/TLS and send insecure data
  - Every connection should use TLS and check certificates; nothing short of this is acceptable
  - Software shouldn't rely on home-grown security mechanisms. Use a reliable layer like TLS.
  - Assume the IP network is insecure; it often is.
  - Sensitive programs shouldn't be used on Wi-Fi

...Case study: insecure application...

- Actually getting hacked is an unlucky combination of network circumstances and software/hardware circumstances



How is a computer hacked?

- Many scenarios, we will focus on one:
  - Computer connected to a network (victim)
  - External attacker also has access to network
    - This could be the Internet, or just a LAN
    - i.e. could be bad guy using Wi-Fi on your LAN
    - Or could be a bad student at the university
  - External attacker knows nothing about victim
  - Attacker wants to gain access, somehow

… How is a computer hacked? ...

- Attackers typically want to know what services this victim has (what IP ports are reachable)
- The “nmap” tool can scan for open IP ports
- This is of interest, because network services often have exploitable bugs
- Those exploits vary greatly on specific cases

… How is a computer hacked? …

- Sample nmap scan output on Windows host
- This victim has open RPC (remote procedure call) and NetBIOS ports, among others

```
> nmap -O 192.168.0.100

Starting Nmap 4.11 ( http://www.insecure.org/nmap/ ) at 2009-02-03 00:28 CST
Interesting ports on 192.168.0.100:
Not shown: 1676 closed ports
PORT     STATE SERVICE
135/tcp  open  msrpc
139/tcp  open  netbios-ssn
445/tcp  open  microsoft-ds
1025/tcp open  NFS-or-IIS
MAC Address: 00:50:BA:CF:07:B7 (D-link)
Device type: general purpose
Running: Microsoft Windows NT/2K/XP
OS details: Microsoft Windows XP Pro SP1/SP2 or 2000 SP4

Nmap finished: 1 IP address (1 host up) scanned in 1.911 seconds
```

… How is a computer hacked? ...

- Each open port represents a service running on the victim computer
- Most services have vulnerable versions
- Search of “windows critical rpc” brought up
  - Microsoft Security Bulletin MS08-067 – Critical
    - “The vulnerability could allow remote code execution if an affected system received a specially crafted RPC request.”
  - Describes an RPC flaw reported October 2008
- If the victim did not update the OS to patch this RPC vulnerability, they are likely exploitable.

… How is a computer hacked? ...

- Many computers run older operating systems and have components that are out of date
- Attacks are not Windows-specific
- Linux, FreeBSD, etc. hosts also run services
  - A host with vulnerable services can be hacked
- The actual exploits usually circulate on the Internet and can do a variety of things
- Typically, an attacker wishes to run a custom program to gain some form of access/control

… How is a computer hacked? ...

- How to minimize risk of getting hacked:
  - Close unnecessary services (ports). Each open service is a potentially vulnerable entry point.
  - Keep software up to date, especially the operating system and services.
  - Restrict access to ports from the outside world, using a firewall.