Security - Networking, part 2

Note Title 3/1/2011

Announcements

- Exam on Thursday
- Scholarships (for juniors + seniors next year)
  apps due in main office
  Stop by 1014 Ritter to get app.
- Post lab 3 at end of week
  due next Friday

Last time: Networking

- IP addresses
- NAT
- Subnetting
- ARP & MAC addresses
- LANs & Network Topologies

The TCP protocol (reliable)

To establish a connection:

1. Client requests a connection by sending a SYN to the server.
2. Server acknowledges by sending a SYN-ACK back to client.
3. Client responds with an ACK, & connection is established.

(TCP 3-way handshake)

Example Attack: SYN flood

Originally, half open connections were stored in a queue that was very short.

$\hookrightarrow$ 8 entries

Once queue was full, connections are ignored until either something times out (~3 min.) or an ACK comes in.

SYN floods were considered unavoidable.

SYN cookies : Daniel J. Bernstein, 1996

"particular choices of initial TCP sequence numbers by TCP servers"

When SYN-ACK is sent, the server attaches a sequence number, n.

This number can be chosen carefully so that the server does not need to store the SYN - it can be reconstructed from the clients ACK (which contains n+1 in it).

Next Lab:

SYN flood simulation
(with & without SYN cookies)

Will use tcpdump — more later.

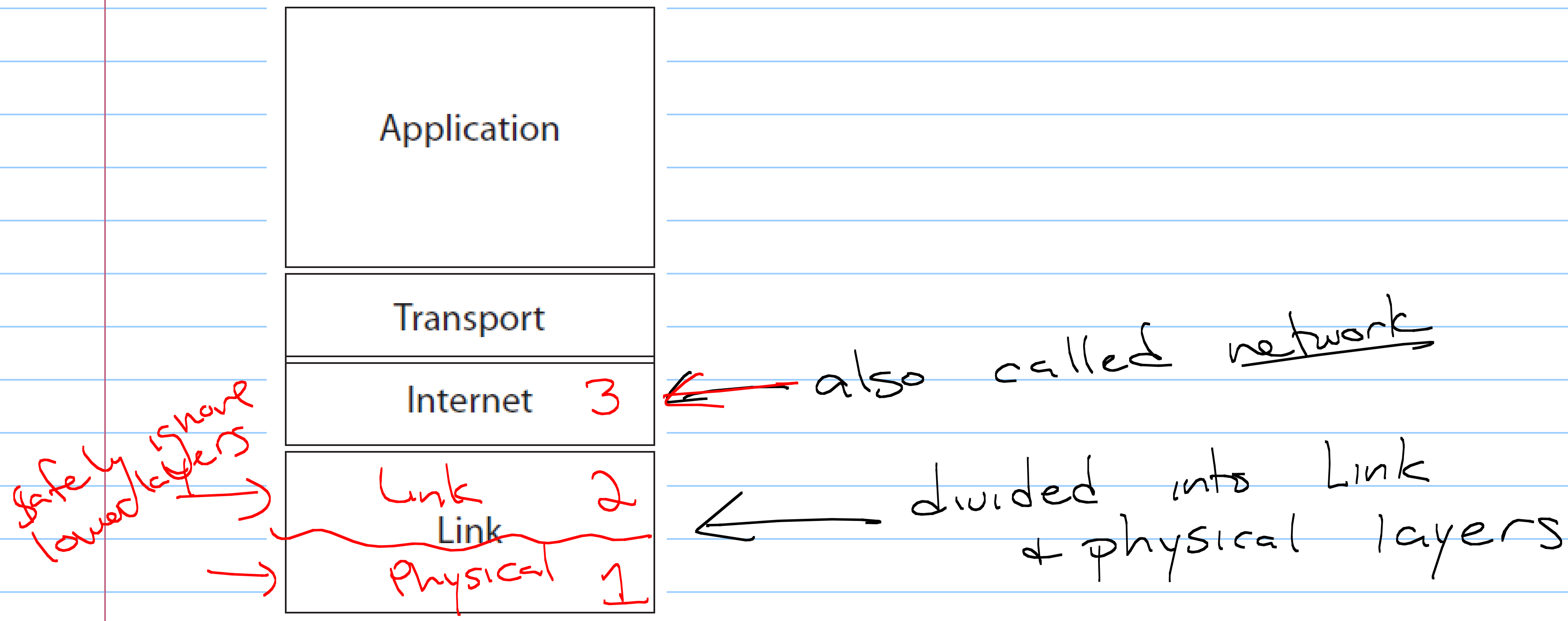
Also Java front end.

Forwarding Devices

Hubs, switches and routers are all forwarding devices.

However, they work differently (and have very different security implications).

Recall the layers:



Hubs

- Layer 1 device, built with knowledge of the physical layer.
- Forwards all frames to all hosts.
- Security is impossible.

Switches

- Layer 2 devices, with knowledge of the link layer.
- Extract MAC address from a packet.
- As a result, the frame is only delivered to the correct destination (as well as any devices on the route there)!

Routers

- Layer 3 devices, with knowledge of network (or internet) layer.

- Able to :

    - perform like switches
    - forward frames across different kinds of networks.
    - forward frames across networks with different Net IDs.
    - utilize NAT

From a black-hat perspective:

Which is better - hubs or routers?

Goal: Turn a router into a hub.

So – how can we make routers more "hub-like" in an attack?

Answer: Use ARP Table against them.
(ARP-poisoning)

Send unsoliated ARP replies to fill up the table, so it has to send ARP requests.

Then reply with your MAC address.

Goal: Convince the router that you
are a different machine.
(Like the default gateway!)

Any traffic is then sent to you instead
of the legitimate goal.

You can then passively sniff, modify
traffic (called man-in-the-middle attack),
or launch denial of service attack.

<u>Defenses</u>: Extra software to check (Many available commercially.)

<u>Note</u>: This can be legitimate!
Ever been redirected to a sign-on page?
Also useful for maintenance of services if a machine needs to be updated or fixed.

How to implement? (lab 4)

Tools: - Tcpdump
(like Wireshark)

Tcp dump is a command line tool that "dumps" all traffic from a network interface.

Must be root or admin!

On a hub network, can see <u>all</u> traffic.

On a switched network, can see all traffic intended for your interface.

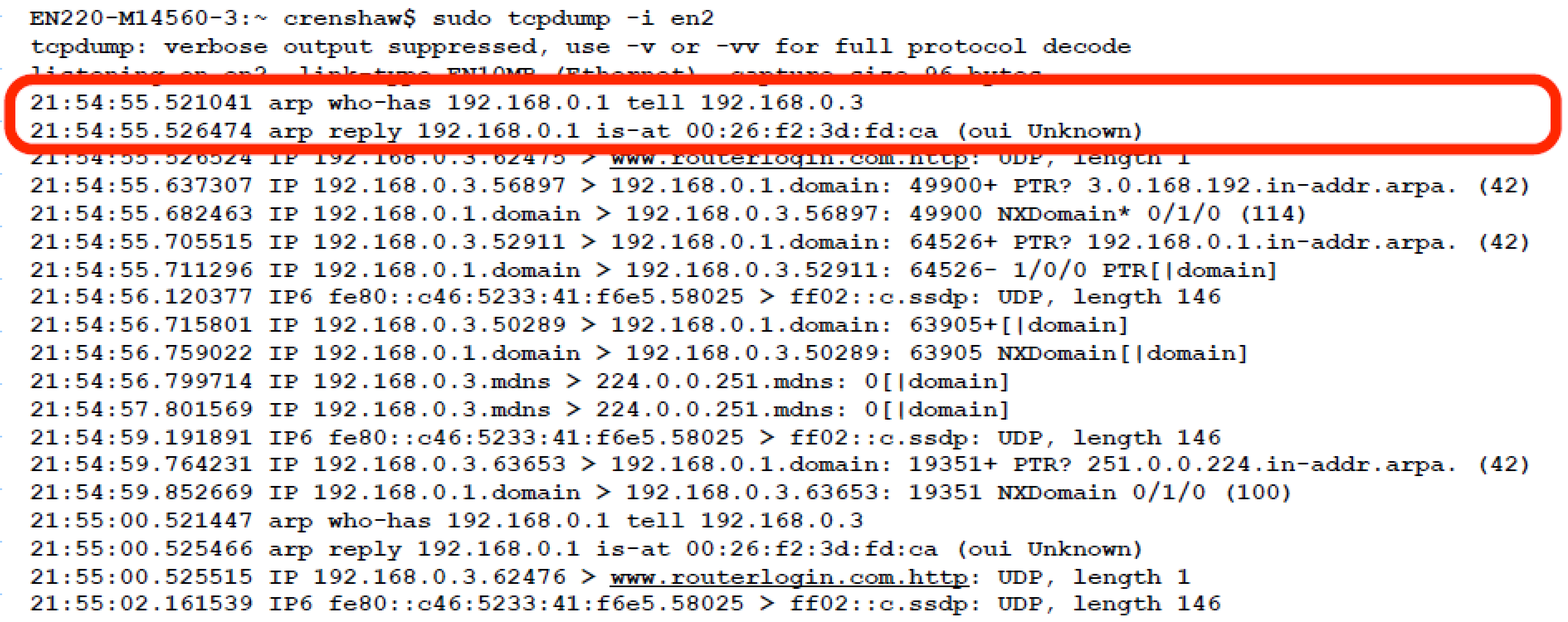
Good tutorial:

http://danielmiessler.com/study/tcpdump/

Example:

```
EN220-M14560-3:~ crenshaw$ ifconfig
...
en2: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
     inet 192.168.0.3 netmask 0xffffff00 broadcast 192.168.0.255
...

EN220-M14560-3:~ crenshaw$ sudo tcpdump -i en2
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on en2, link-type EN10MB (Ethernet), capture size 96 bytes
21:54:55.521041 arp who-has 192.168.0.1 tell 192.168.0.3
21:54:55.526474 arp reply 192.168.0.1 is-at 00:26:f2:3d:fd:ca (oui Unknown)
21:54:55.526524 IP 192.168.0.3.62475 > www.routerlogin.com.http: UDP, length 1
21:54:55.637307 IP 192.168.0.3.56897 > 192.168.0.1.domain:  49900+ PTR? 3.0.168.192.in-addr.arpa. (42)
21:54:55.682463 IP 192.168.0.1.domain > 192.168.0.3.56897:  49900 NXDomain* 0/1/0 (114)
21:54:55.705515 IP 192.168.0.3.52911 > 192.168.0.1.domain:  64526+ PTR? 192.168.0.1.in-addr.arpa. (42)
21:54:55.711296 IP 192.168.0.1.domain > 192.168.0.3.52911:  64526- 1/0/0 PTR[|domain]
21:54:56.120377 IP6 fe80::c46:5233:41:f6e5.58025 > ff02::c.ssdp: UDP, length 146
21:54:56.715801 IP 192.168.0.3.50289 > 192.168.0.1.domain:  63905+[|domain]
21:54:56.759022 IP 192.168.0.1.domain > 192.168.0.3.50289:  63905 NXDomain[|domain]
21:54:56.799714 IP 192.168.0.3.mdns > 224.0.0.251.mdns:  0[|domain]
21:54:57.801569 IP 192.168.0.3.mdns > 224.0.0.251.mdns:  0[|domain]
21:54:59.191891 IP6 fe80::c46:5233:41:f6e5.58025 > ff02::c.ssdp: UDP, length 146
21:54:59.764231 IP 192.168.0.3.63653 > 192.168.0.1.domain:  19351+ PTR? 251.0.0.224.in-addr.arpa. (42)
21:54:59.852669 IP 192.168.0.1.domain > 192.168.0.3.63653:  19351 NXDomain 0/1/0 (100)
21:55:00.521447 arp who-has 192.168.0.1 tell 192.168.0.3
21:55:00.525466 arp reply 192.168.0.1 is-at 00:26:f2:3d:fd:ca (oui Unknown)
21:55:00.525515 IP 192.168.0.3.62476 > www.routerlogin.com.http: UDP, length 1
21:55:02.161539 IP6 fe80::c46:5233:41:f6e5.58025 > ff02::c.ssdp: UDP, length 146
```



```
EN220-M14560-3:~ crenshaw$ sudo tcpdump -i en2
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on en2, link-type EN10MB (Ethernet), capture size 96 bytes
21:54:55.521041 arp who-has 192.168.0.1 tell 192.168.0.3
21:54:55.526474 arp reply 192.168.0.1 is-at 00:26:f2:3d:fd:ca (oui Unknown)
21:54:55.526524 IP 192.168.0.3.62475 > www.routerlogin.com.http: UDP, length 1
21:54:55.637307 IP 192.168.0.3.56897 > 192.168.0.1.domain: 49900+ PTR? 3.0.168.192.in-addr.arpa. (42)
21:54:55.682463 IP 192.168.0.1.domain > 192.168.0.3.56897: 49900 NXDomain* 0/1/0 (114)
21:54:55.705515 IP 192.168.0.3.52911 > 192.168.0.1.domain: 64526+ PTR? 192.168.0.1.in-addr.arpa. (42)
21:54:55.711296 IP 192.168.0.1.domain > 192.168.0.3.52911: 64526- 1/0/0 PTR[|domain]
21:54:56.120377 IP6 fe80::c46:5233:41:f6e5.58025 > ff02::c.ssdp: UDP, length 146
21:54:56.715801 IP 192.168.0.3.50289 > 192.168.0.1.domain: 63905+[|domain]
21:54:56.759022 IP 192.168.0.1.domain > 192.168.0.3.50289: 63905 NXDomain[|domain]
21:54:56.799714 IP 192.168.0.3.mdns > 224.0.0.251.mdns: 0[|domain]
21:54:57.801569 IP 192.168.0.3.mdns > 224.0.0.251.mdns: 0[|domain]
21:54:59.191891 IP6 fe80::c46:5233:41:f6e5.58025 > ff02::c.ssdp: UDP, length 146
21:54:59.764231 IP 192.168.0.3.63653 > 192.168.0.1.domain: 19351+ PTR? 251.0.0.224.in-addr.arpa. (42)
21:54:59.852669 IP 192.168.0.1.domain > 192.168.0.3.63653: 19351 NXDomain 0/1/0 (100)
21:55:00.521447 arp who-has 192.168.0.1 tell 192.168.0.3
21:55:00.525466 arp reply 192.168.0.1 is-at 00:26:f2:3d:fd:ca (oui Unknown)
21:55:00.525515 IP 192.168.0.3.62476 > www.routerlogin.com.http: UDP, length 1
21:55:02.161539 IP6 fe80::c46:5233:41:f6e5.58025 > ff02::c.ssdp: UDP, length 146
```

Main-in-the-Middle Attacks

We'll briefly discuss several types of man-in-the-middle attacks!

(This will also be a lab later on!)

ARP Poisoning

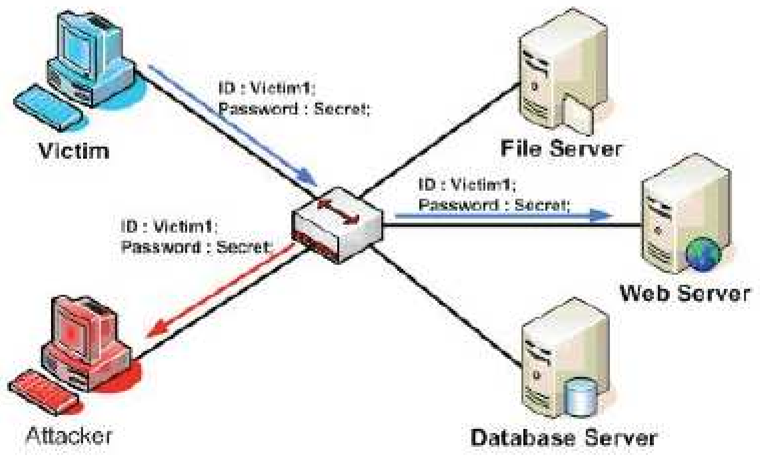


image taken from: http://www.owasp.org/index.php/Network_Eavesdropping

Recall key exchanges

[1] Alice sends her public key to Bob.

[2] Bob sends his public key to Alice.

[3] When Alice sends a message to Bob, it is encrypted with Bob's public key. Bob can decrypt the messages with his private key.

[4] . When Bob sends a message to Alice, it is encrypted with Alice's public key. Alice can decrypt the messages with her private key.

An Attack:

[1] Alice sends her public key to Bob. Mallory intercepts this key and sends Bob his own public key.

[2] Bob sends his public key to Alice. Mallory intercepts this key and sends Alice his own public key.

[3] When Alice sends a message to Bob, Mallory intercepts it. Since this message is really encrypted with his own private key, he decrypts it with his private key, re-encrypts it with Bob's public key and sends it to bob.

[4] Same thing goes when Bob sends a message to Alice.

Example taken from *Applied Cryptography* by Schneier

"A common cryptographic technique is to encrypt each individual conversation with a session key."

--Applied Cryptography by Schneier

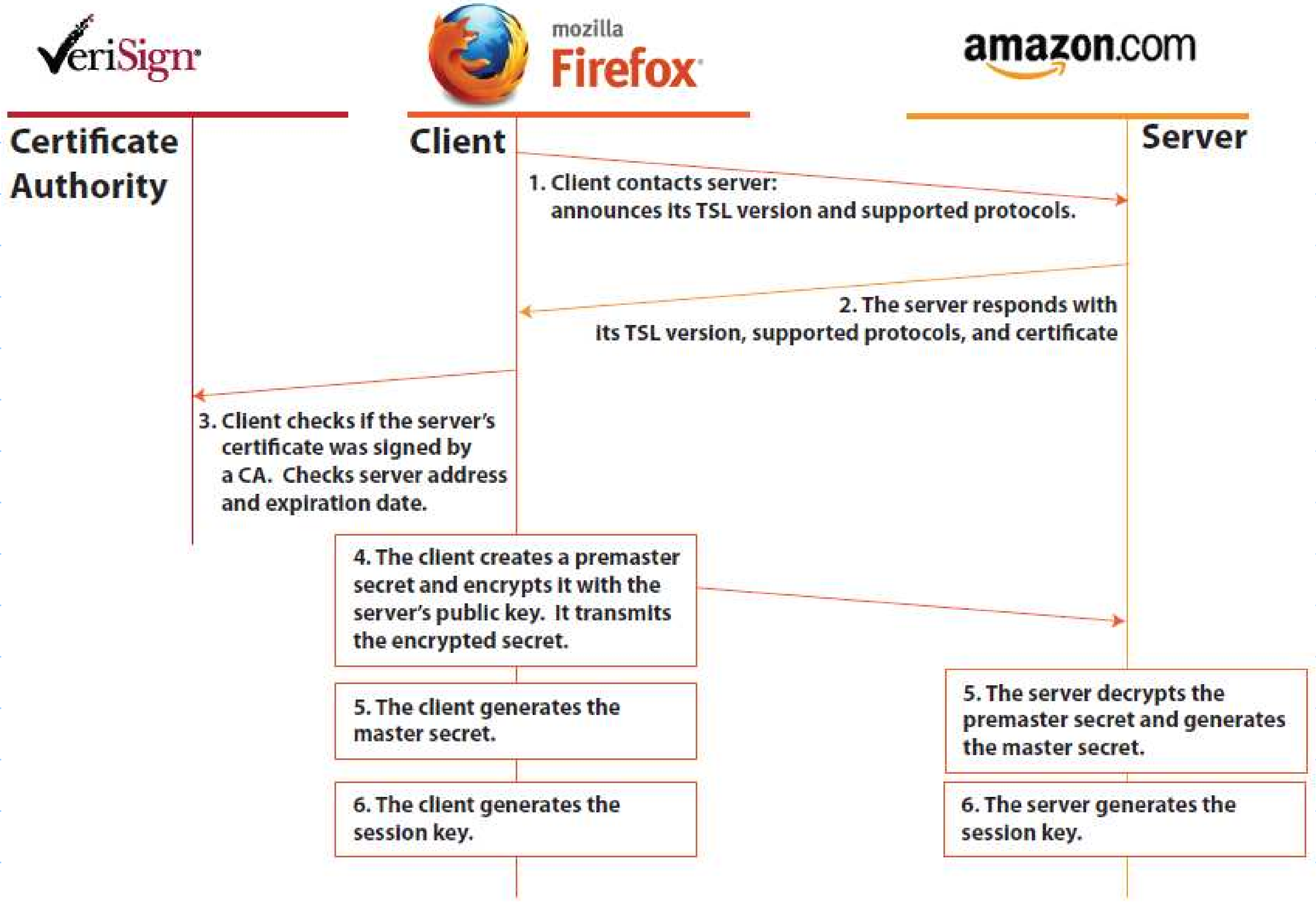
simple key exchange approach

[1] Alice gets Bob's public key from a key distribution center.

[2] Alice generates a random session key, encrypts it using Bob's public key, then sends it to Bob.

[3] Bob then decrypt's Alice's message using his private key.

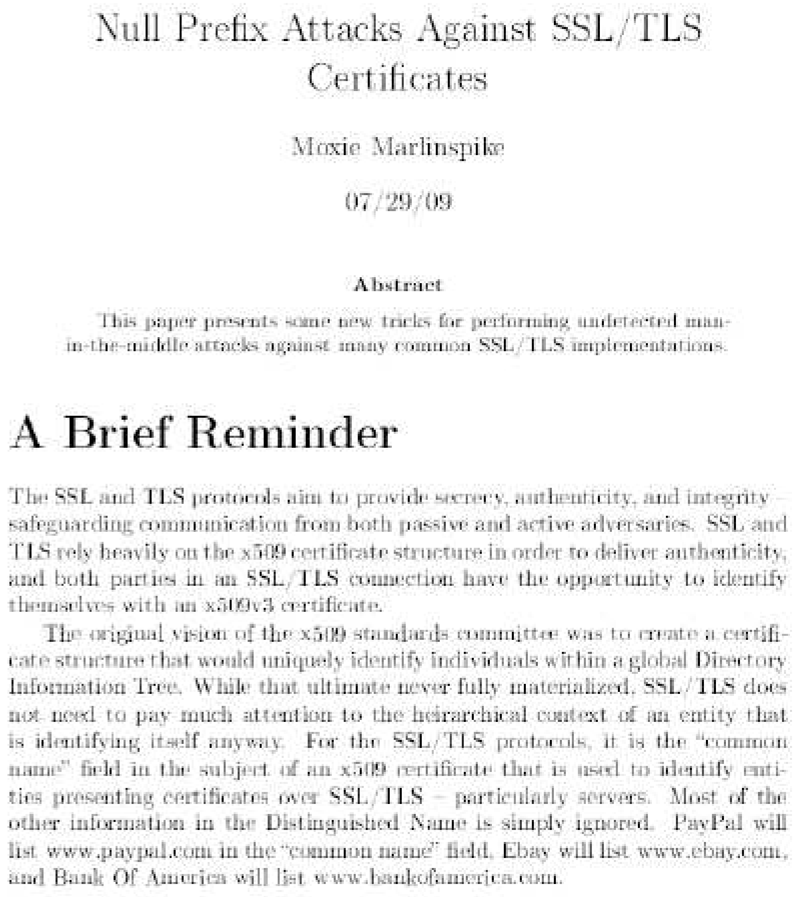
[4] Both of them encrypt messages with the session key.


VeriSign
mozilla Firefox
amazon.com
Certificate Authority
Client
Server
1. Client contacts server: announces its TSL version and supported protocols.
2. The server responds with its TSL version, supported protocols, and certificate
3. Client checks if the server's certificate was signed by a CA. Checks server address and expiration date.
4. The client creates a premaster secret and encrypts it with the server's public key. It transmits the encrypted secret.
5. The client generates the master secret.
6. The client generates the session key.
5. The server decrypts the premaster secret and generates the master secret.
6. The server generates the session key.

Another Attack:

Black Hat DC 2009

Moxie Marlinspike finds a security hole in one kind of certificate used in the SSL and TLS protocols.



Null Prefix Attacks Against SSL/TLS Certificates

Moxie Marlinspike

07/29/09

Abstract

This paper presents some new tricks for performing undetected man-in-the-middle attacks against many common SSL/TLS implementations.

A Brief Reminder

The SSL and TLS protocols aim to provide secrecy, authenticity, and integrity – safeguarding communication from both passive and active adversaries. SSL and TLS rely heavily on the x509 certificate structure in order to deliver authenticity, and both parties in an SSL/TLS connection have the opportunity to identify themselves with an x509v3 certificate.

The original vision of the x509 standards committee was to create a certificate structure that would uniquely identify individuals within a global Directory Information Tree. While that ultimate never fully materialized, SSL/TLS does not need to pay much attention to the heirarchical context of an entity that is identifying itself anyway. For the SSL/TLS protocols, it is the "common name" field in the subject of an x509 certificate that is used to identify entities presenting certificates over SSL/TLS – particularly servers. Most of the other information in the Distinguished Name is simply ignored. PayPal will list www.paypal.com in the "common name" field, Ebay will list www.ebay.com, and Bank Of America will list www.bankofamerica.com.

Useful Tricks

x509 certificates are formatted using ASN.1 notation. ASN.1 supports many different string types, but all of them are represented as some variation of PASCAL strings. In memory, PASCAL strings are represented by a series of bytes which specify the length of the string, followed by the string data itself – one character per byte. This is in contrast to C strings, which are represented in memory as a series of bytes – one character per byte – which are terminated by a single NULL character.

PASCAL String:

0x04 (Length)	0x44 ('D')	0x41 ('A')	0x54 ('T')	0x41 ('A')

C String:

0x44 ('D')	0x41 ('A')	0x54 ('T')	0x41 ('A')	0x00 (NULL)

One important effect of representing strings in PASCAL format is that NULL characters are treated just like any other character in your character string and are not embued with any special meaning. This means that we can freely include NULL characters in any of the fields within our x509 certificates, including the "common name" field.

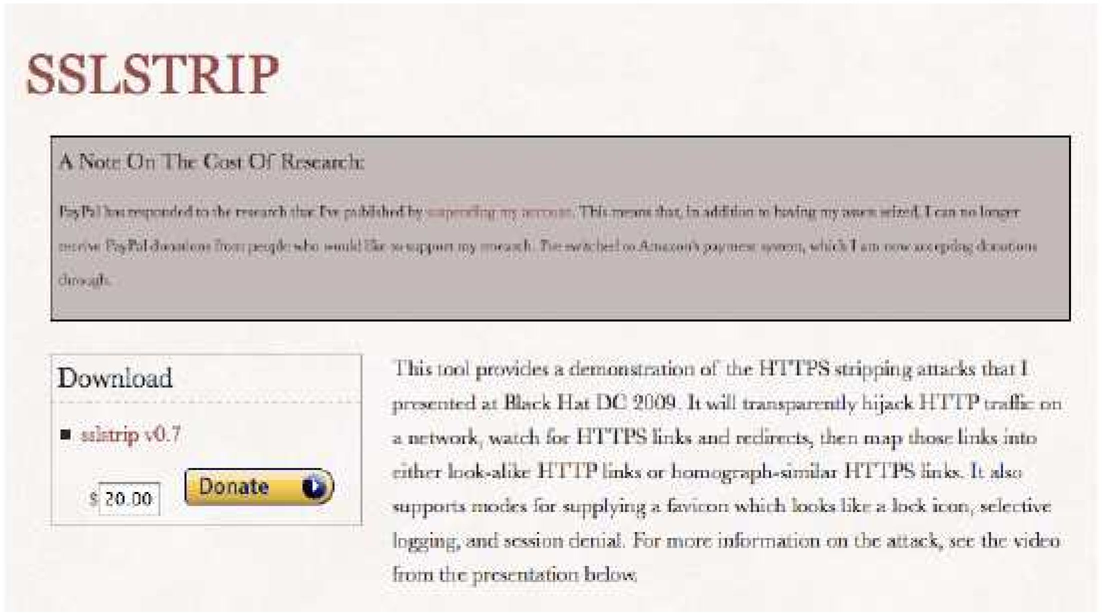
One might issue a Certificate Signing Request like this:

www.paypal.com\0.thoughtcrime.org

As mentioned, the Certificate Authority will ignore the prefix, and only examine the root domain, thoughtcrime.org. If the person issuing the request is the legitimate owner of thoughtcrime.org (and presumably he would be), he would be able to prove his ownership of the domain to the Certificate Authority without any difficulty.

As it stands, most contemporary SSL/TLS implementations treat the fields obtained from x509 certificates as ordinary C strings, using ordinary C string functions for comparison and manipulation. As a consequence of this, a string comparison between www.paypal.com\0.thoughtcrime.org and www.paypal.com will identify the two strings as identical. The owner of the certificate for www.paypal.com\0thoughtcrime.org can thus successfully present this certificate for connections intended to www.paypal.com, effectively defeating the authenticity property of SSL/TLS and allowing for, among other things, undetectable man-in-the-middle attacks.

Current state of affairs:


SSLSTRIP

A Note On The Cost Of Research:

PayPal has responded to the research that I've published by suspending my account. This means that, in addition to having my assets seized, I can no longer receive PayPal donations from people who would like to support my research. I've switched to Amazon's payment system, which I am now accepting donations through.

Download

- sslstrip v0.7

$ 20.00 Donate

This tool provides a demonstration of the HTTPS stripping attacks that I presented at Black Hat DC 2009. It will transparently hijack HTTP traffic on a network, watch for HTTPS links and redirects, then map those links into either look-alike HTTP links or homograph-similar HTTPS links. It also supports modes for supplying a favicon which looks like a lock icon, selective logging, and session denial. For more information on the attack, see the video from the presentation below.

Today, he's selling a tool called sslstrip which exploits this hole.

http://www.thoughtcrime.org/software/sslstrip/