Greg Niemeyer

HW03

1.

a) 1) [www.amazon.com](http://www.amazon.com), [www.baidu.com](http://www.baidu.com), [www.madisonproperty.com](http://www.madisonproperty.com)

2)

Amazon: adventures in Stochastic Processes

<http://www.amazon.com/s/ref=nb_sb_noss_1?url=search-alias%3Dstripbooks&field-keywords=adventures+in+Stochastic+Processes>

Baidu: 来自星星的

<http://www.baidu.com/s?wd=%E6%9D%A5%E8%87%AA%E6%98%9F%E6%98%9F%E7%9A%84%E4%BD>

Bing: madison map

<https://www.bing.com/search?q=madison+map&go=&qs=n&form=QBLH&pq=madison+map&sc=8-5&sp=-1&sk=&cvid=21ab45dc55634583bed819be11051864>

b) 1) user: shiningmoon, pass: public

2) In active mode, the client initiates opening the command channel with the server and the server initiates opening the data channel with the client. In client mode, the client initiates opening both of the channels with the server.

3) No active mode connections, since any time the data channel was used it was set up in passive mode

4) 30-46, 75-99, 110-126, 151-176, 181-200, 218-233, 244-263 => passive mode data channel setup and teardown.

5) dragon.zip, ARP.java, L2Switch.java, phase1.html

c) 1) Source: 198.168.0.100

2) Destination 74.125.225.46

3) Route:

198.168.0.1

10.131.180.1

96.34.20.20

96.34.17.95

96.34.16.112

96.34.16.77

96.34.2.4

96.34.0.7

96.34.0.9

96.34.152.30

209.85.154.120

209.85.250.28

d) 1) [cs155@dummymail.com](mailto:cs155@dummymail.com) whitehat

2) 5 messages

3) 1) from: cs155@dummymail.com

to: cs155@dummymail.com

subject: foobar

date: 4/23/10 8:20:55

2) from: cs155@dummymail.com

to: cs155@dummymail.com

subject: can you see this subject?

date: 4/23/10 8:23:26

3) from: harinym@stanford.edu

to: cs155@dummymail.com

subject: test message

date: 4/23/10 10:25:00

4) from: harinym@stanford.edu

to: cs155@dummymail.com

subject: wassup

date: 4/23/10 8:21:50

5) from: harinym@stanford.edu

to: cs155@dummymail.com

subject: geology rocks!

date: 4/23/10 8:22:29

2.

a) If the packets are simply being logged for later inspection, and attack will not be prevented, only recognized at best. A SYN flood could still be used to take down the server and the IDS.

4.

a) SSL protects against man in the middle attacks because an attacker cannot impersonate a server because the server has a digital signature certified by a trusted CA that the client can check. The server also sends a nonce and its public key with the digital signature, so that the key can be assured to be the server’s. The client sends back a “pre-master’s secret” to the server encrypted under its public key and then a session key can be generated that the server and the client only know.

b) SSL protects against password sniffing because once the connection is established and the session key has been generated, all communication between client and server is encrypted with the session key, which only the client and server know and only they can decrypt these messages. Without the ability to decrypt the password packet, the packet will not make any sense to the eavesdropper.

c) SSL protects against a IP spoofing because the SSL certificate is specific to domain, not IP, so the spoofed IP will not have the credentials to fool the host provided the host checks the certificate.

d) SSL protects against IP hijacking because even if the attacker takes the place of one of the hosts, the communication is still encrypted with a session key only the original host and server know.

e) SSL does not protect against SYN flooding since the attack happens at the TCP level which is the level below SSL.

f) SSL protects against replay attacks because during the SSL handshake, both the client and server send nonces to each other, which are unique random numbers. The replayed messages will be of no use to the attacker since the server will use a new nonce hashed with the messages and the two will be unable to communicate.

g) SSL protects against known plain-text attacks because the text is compressed (which increases entropy by removing redundancy), then prepended to the MAC and encrypted, making the text unpredictable.

5.

a) An attacker can inject his own specially crafted code into a website’s HTML if that website contains a XSS vulnerability. He can inject code that executes in the victims browser every time he visits the site with the malicious script embedded in it, including a code that steals cookies from the victims browser and log them somewhere the attacker can view them. Alternatively, he can temporarily inject the code, send the malicious link to the unsuspecting user and steal his cookies that way.

b) If the victim forgets to log out of his bank website, and the bank website does not automatically log him out when he leaves the page, an attacker can craft an attack that impersonates the victim and resets his password to anything the attacker wants without the victims approval. This works by tricking the victim into unwittingly sending the server an HTTP form submission with fields the attacker has chosen. When the form submits, the server thinks the victim is performing this request because his cookie has not expired and he is still logged in. The site could check the referrer header in the HTTP, which contains the last page the user was on before the request was sent. If the referrer was not from the correct page inside the domain, the request should be rejected.

c)

* XSRF is a confused deputy attack because the server thinks that it is acting on behalf of the victim, but it is actually carrying out the whim of the attacker.
* The server is the confused deputy in the XSRF attack.
* The server is acting on the principles that since the user’s cookie is still valid, the server should perform requests on his behalf.

d) By checking the referrer header in the HTTP request, you ensure that the user came from a page inside the domain, and not from some other site. The fact that the referrer came from inside the domain acts as a capability in the fact that since the request came from a source that only the specific user has access too and not some outside page, the referrer grants the capability to change the password.

e) setuid programs in UNIX are an example of confused deputies because the program is given all the privileges of the root. These programs can be maliciously exploited to perform actions as the root on behalf of an attacker.