

KillJoy 0.1

Mobile TCP Session Management

Cameron Flint, Hope College, Spring 2010



Motivation

- Free up resources... on a network I don't necessarily own
 - Bittorrent traffic
 - Large downloads
 - Media streaming
 - Whatever

Motivation

- Desire to control traffic over TCP on a network without any escalated privileges
 - Ability to monitor all sessions for all local IPs on the LAN, passively
 - Ability to terminate a connection
- Desire for automation / portability
 - Maintain a blacklist (by IP or session)
- Desire to determine the type of observed traffic
 - Higher-level protocol detection

Implementation

- Tools and toolkits
 - Python 2.5
 - Scapy
 - GTK (+hildon)
- Development environments (mixed results)
 - Scratchbox
 - Eclipse + ESBox, Pluthon, Pydev, Mica, etc.
 - **SSH and SCP**

Implementation

- Terminating sessions
 - Fakes a few TCP RST packets:
 - Receive packet from server, p1
 - Send response, p2, with the following field configuration, back to the server:
 - `p2[IP].src=p1[IP].dst` # fake source
 - `p2[IP].dst=p1[IP].src` # send to right addr
 - `p2[TCP].seq=p1[TCP].ack` # hijack session sequence
 - `p2[TCP].sport=p1[TCP].dport` # fake source port
 - `p2[TCP].dport=p1[TCP].sport` # send to right port
 - Check for continuation
- ARP ping / cache poisoning
 - Nearly identical to Scapy's `arping()` and `arpcachepoison()`

Demo

Challenges

- True promiscuous mode not available?
 - Eventually realized that neither handheld device, nor my laptop, was able to report network traffic other than broadcasted traffic, or what it sent itself
 - How can this be the case on a wireless router?? (temporary descent into mad confusion)

Workaround: *Provide ARP scan and redirect features.*

Current State

- Works on *my* network...
- **Higher protocol recognition NOT implemented**
- Cannot *prevent* a connection from being made
- Automation is a bit noisy

Thanks for watching!