Thieves can hijack $28,000 professional drones used widely across the law enforcement, emergency, and private sectors using $40 worth of hardware. The quadcopters can be hijacked from up to two kilometers away thanks to a lack of encryption, which is not present due to latency overheads.

Attackers can commandeering radio links to the drones from up to two kilometers away, and block operators from reconnecting to the craft. With the targeted Xbee chip being very common in drones, IBM security guy Nils Rodday says it is likely many more aircraft are open to compromise.
network security

adam everspaugh
ace@cs.wisc.edu
today

* **Announcement:** HW3 to be released
* WiFi
* IP, TCP
* DoS, DDoS, prevention
802.11 (wifi)

STA = station
AP = access point

BSS = basic service set
DS = distribution service
ESS = extended service set

SSID (service set identifier)
identifies the 802.11 network

Typical WiFi modes:
Unsecured
Wireless Protected Access (WPA2) - password authenticated, encrypted

802.11 association

Probe request
SSID: “linksys”, BSSID: MAC1
Auth request MAC1
Auth response
Associate request MAC1
Associate response
Two APs for same network

Probe request
SSID: “linksys”, BSSID: MAC1
SSID: “linksys”, BSSID: MAC2

Auth request MAC2

Choose one of MAC1, MAC2
802.11 evil twins

Basic idea:
- Attacker pretends to be an AP to intercept traffic or collect data

Basic attack: rogue AP

Probe request
SSID: “linksys”, BSSID: MAC1
SSID: “linksys”, BSSID: MAC2

Auth request MAC2

Choose one of MAC1, MAC2

What if client choose MAC1?
Attacker may try to send a forged reset message and force re-connect
Parrot ARdrone

Drone is a WiFi access point
Uses unsecured 802.11 connection (WiFi)
Controlled from iPad or iPhone with an app
Uses MAC address for security
Internet protocol stack

Application
TCP
IP
Ethernet

TCP segment
IP datagram
Ethernet frame

14 20 20
46 to 1500 bytes
IP protocol (IPv4)

• Connectionless
  – no state
• Unreliable
  – no guarantees
• ICMP (Internet Control Message Protocol)
  – often used by tools such as ping, traceroute
## IPv4

<table>
<thead>
<tr>
<th>ENet hdr</th>
<th>IP hdr</th>
<th>data</th>
<th>ENet tlr</th>
</tr>
</thead>
</table>

An Ethernet frame containing an IP datagram consists of:

<table>
<thead>
<tr>
<th>4-bit version</th>
<th>4-bit hdr len</th>
<th>8-bit type of service</th>
<th>16-bit total length (in bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>16-bit identification</td>
<td>3-bit flags</td>
<td>13-bit fragmentation offset</td>
<td></td>
</tr>
<tr>
<td>8-bit time to live (TTL)</td>
<td>8-bit protocol</td>
<td>16-bit header checksum</td>
<td></td>
</tr>
<tr>
<td>32-bit source IP address</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>32-bit destination IP address</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>options (optional)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Security issues with IP

Routing has issues, we’ll get to that later
What else?
- No source address authentication in general
Denial of Service (DoS) attacks

Goal is to prevent legitimate users from accessing victim (1.2.3.4)

ICMP ping flood
- Attacker sends ICMP pings as fast as possible to victim
- When will this work as a DoS?  Attacker resources > victim’s
- How can this be prevented?  Ingress filtering near victim

think-pair-share
Denial of Service (DoS) attacks

How can attacker avoid ingress filtering?

Attacker can send packet with fake source IP (*packet spoofing*)
Packet will get routed correctly
Replies will not

Send IP packet with
source: 8.7.3.4
dest: 1.2.3.4
from 5.6.7.8

Filter based on source may be incorrect
DoS reflection attacks

Note: echo request, DEST IP=8.7.3.4, SRC IP=1.2.3.4
- Attacker can bounce an attack against 1.2.3.4 off 8.7.3.4
- Avoid source filtering
Denial of Service (DoS) attacks

DoS works best when there is **asymmetry** between victim and attacker
- Attacker uses few resources to cause victim to consume lots of resources
DoS works best when there is asymmetry between victim and attacker

Example: DNS reflection attacks
Send DNS request with spoofed source IP (~65 byte request)
DNS replies sent to target (~512 byte response)
Reflect + amplify the attack
Estonia attack

Distributed DoS (DDoS)
• April 2007
• Used army of bots
• Attacks continued for weeks with varying intensities
• Targeted government, banks, news, university web sites

[ATLAS 2007]
From analysis of 2 weeks of attack traffic
• 120+ distinct attacks
• 115 ICMP floods, 4 TCP SYN floods
• 12 attacks: 70-95 Mbps for 10+ hrs
• All attack traffic from outside Estonia
• Solution: Block all foreign traffic until attacks subsided
Internet protocol stack

<table>
<thead>
<tr>
<th>Layer</th>
<th>Protocols/Standards</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>HTTP, FTP, SMTP, SSH, etc.</td>
</tr>
<tr>
<td>Transport</td>
<td>TCP, UDP</td>
</tr>
<tr>
<td>Network</td>
<td>IP, ICMP, IGMP</td>
</tr>
<tr>
<td>Link</td>
<td>802x (802.11, Ethernet)</td>
</tr>
</tbody>
</table>
TCP (transport control protocol)

• Connection-oriented
  – state initialized during handshake and maintained

• Goal: reliable, ordered, error-checked delivery of a stream of bytes
  – generates segments
  – timeout segments that aren’t acknowledged
  – reorders received segments when necessary
TCP (transport control protocol)

<table>
<thead>
<tr>
<th></th>
<th>IP hdr</th>
<th>TCP hdr</th>
<th>data</th>
</tr>
</thead>
<tbody>
<tr>
<td>16-bit</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>source port number</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16-bit</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>destination port number</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>32-bit</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>sequence number</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>32-bit</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>acknowledgement number</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4-bit hdr len</td>
<td>6-bits</td>
<td>6-bits</td>
<td>16-bit</td>
</tr>
<tr>
<td>reserved</td>
<td>flags</td>
<td></td>
<td>window size</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16-bit</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TCP checksum</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16-bit</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>urgent pointer</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>options (optional)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>data (optional)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
TCP (transport control protocol)

TCP flags

<table>
<thead>
<tr>
<th>Flag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>URG</td>
<td>urgent pointer valid</td>
</tr>
<tr>
<td>ACK</td>
<td>acknowledgement number valid</td>
</tr>
<tr>
<td>PSH</td>
<td>pass data to app ASAP</td>
</tr>
<tr>
<td>RST</td>
<td>reset connection</td>
</tr>
<tr>
<td>SYN</td>
<td>synchronize sequence #’s</td>
</tr>
<tr>
<td>FIN</td>
<td>finished sending data</td>
</tr>
</tbody>
</table>
TCP handshake

Client

SYN seqC,0

SYN/ACK seqS,seqC+1

ACK seqC+1,seqS+1

TCP connection established

SYN = syn flag set
ACK = ack flag set
x,y = x is sequence #, y is acknowledge #
TCP SYN floods

Send lots of **TCP SYN** packets to 1.2.3.4, no **ACK**
- 1.2.3.4 maintains state for each **SYN** packet for some time window
  - What **asymmetry** is being abused?
  - What **SRC IP** does attacker use?
  - If attackers sets SRC IP=8.7.3.4, what does 8.7.3.4 receive?
Preventing DDoS

Large number of front-end servers absorb traffic
Forward legitimate-looking traffic to back-end servers

Companies and web sites pay for this: CloudFlare, Arbor Networks, Akamai, and many others
recap

* WiFi Evil Twins

* DoS
  / ICMP Flood
  / DDoS
  / DNS reflection, amplification
  / TCP SYN Flooding
  / Preventing DDoS

* Exit slips
  / 1 thing you learned
  / 1 thing you didn't understand