Network Security

Security in local-area networks

Radboud University, The Netherlands

Spring 2019

A short recap

- Ethernet takes care of transporting frames on the *link layer*
- Nodes on the link layer are identified by 48-bit MAC (“physical”) addresses

A short recap

- Ethernet takes care of transporting frames on the *link layer*
- Nodes on the link layer are identified by 48-bit MAC (“physical”) addresses
- IP addresses are more flexible logical addresses used on the *internet layer*
- Mapping from MAC addresses to IP addresses is the job of ARP
- Hostnames are aliases for IP addresses

A short recap

- Ethernet takes care of transporting frames on the *link layer*
- Nodes on the link layer are identified by 48-bit MAC (“physical”) addresses
- IP addresses are more flexible logical addresses used on the *internet layer*
- Mapping from MAC addresses to IP addresses is the job of ARP
- Hostnames are aliases for IP addresses
- End-to-end communication is handled by TCP
- TCP *ports* identify communication endpoints (together with the IP address)

A short recap

- Ethernet takes care of transporting frames on the *link layer*
- Nodes on the link layer are identified by 48-bit MAC ("physical") addresses
- IP addresses are more flexible logical addresses used on the *internet layer*
- Mapping from MAC addresses to IP addresses is the job of ARP
- Hostnames are aliases for IP addresses
- End-to-end communication is handled by TCP
- TCP *ports* identify communication endpoints (together with the IP address)
- TCP establishes connections (3-way handshake)
- TCP manages retransmission, ordering of packets etc.

A short recap

- Ethernet takes care of transporting frames on the *link layer*
- Nodes on the link layer are identified by 48-bit MAC ("physical") addresses
- IP addresses are more flexible logical addresses used on the *internet layer*
- Mapping from MAC addresses to IP addresses is the job of ARP
- Hostnames are aliases for IP addresses
- End-to-end communication is handled by TCP
- TCP *ports* identify communication endpoints (together with the IP address)
- TCP establishes connections (3-way handshake)
- TCP manages retransmission, ordering of packets etc.
- UDP much more lightweight, essentially only implements endpoints (ports)

A short recap

- Ethernet takes care of transporting frames on the *link layer*
- Nodes on the link layer are identified by 48-bit MAC ("physical") addresses
- IP addresses are more flexible logical addresses used on the *internet layer*
- Mapping from MAC addresses to IP addresses is the job of ARP
- Hostnames are aliases for IP addresses
- End-to-end communication is handled by TCP
- TCP *ports* identify communication endpoints (together with the IP address)
- TCP establishes connections (3-way handshake)
- TCP manages retransmission, ordering of packets etc.
- UDP much more lightweight, essentially only implements endpoints (ports)
- ICMP is used for various control and administration messages

Sniffing traffic – the good old days

- Can connect two computers directly by Ethernet cable (old installations need a *crossover cable*)
- For > 2 computers need some joint connection

Sniffing traffic – the good old days

- Can connect two computers directly by Ethernet cable (old installations need a *crossover cable*)
- For > 2 computers need some joint connection
- A *hub* (active hub) repeats bits received on one port on all other ports
- Simplified view: all connected Ethernet cables "soldered together"

Sniffing traffic – the good old days

- Can connect two computers directly by Ethernet cable (old installations need a *crossover cable*)
- For > 2 computers need some joint connection
- A *hub* (active hub) repeats bits received on one port on all other ports
- Simplified view: all connected Ethernet cables "soldered together"
- `cersei` plugs computer into the hub:
  - Can listen (sniff, eavesdrop) to all communication between `arya` and `tyrion`
  - Can jam all communication between `arya` and `tyrion` (DOS)
  - Can impersonate `tyrion` or `arya` (more later)

Switched Ethernet

"Hubs are now largely obsolete, having been replaced by network switches except in very old installations or specialized applications." *—Wikipedia article "Ethernet hub"*

- Disadvantage of hubs: lots of frame collisions in busy networks
- Idea: only forward bits to the actual receiver(s) of the frames

Switched Ethernet

"Hubs are now largely obsolete, having been replaced by network switches except in very old installations or specialized applications." —*Wikipedia article "Ethernet hub"*

- Disadvantage of hubs: lots of frame collisions in busy networks
- Idea: only forward bits to the actual receiver(s) of the frames
- *Network switches* are aware of MAC addresses sitting behind their ports ("port" here only refers to a physical plug socket, not a TCP port)
- Forward data only to the port with the receiver MAC address
- Switched Ethernet creates separate *collision domains* for each port

Switched Ethernet

> *"Hubs are now largely obsolete, having been replaced by network switches except in very old installations or specialized applications."* —*Wikipedia article "Ethernet hub"*

- Disadvantage of hubs: lots of frame collisions in busy networks
- Idea: only forward bits to the actual receiver(s) of the frames
- *Network switches* are aware of MAC addresses sitting behind their ports ("port" here only refers to a physical plug socket, not a TCP port)
- Forward data only to the port with the receiver MAC address
- Switched Ethernet creates separate *collision domains* for each port
- Switched Ethernet also creates separate "sniffing domains" for each port
- How about our nice attacks, do they still work?

Put back the hub

Before the attack

```
                _________
              |         |
arya----------|         |
              |         |
tyrion--------| Switch  |
              |         |
cersei--------|         |
              |_________|
```

After the attack

```
                  ________
                 |        |
arya-------------|        |
         _____   |        |
        |     |  | Switch |
tyrion--| Hub |--|        |
        |_____|  |        |
           |     |________|
cersei- ---*
```

Put back the hub

Before the attack

```
                _______
               |       |
arya----------|       |
               |       |
tyrion--------| Switch |
               |       |
cersei--------|       |
               |_______|
```

After the attack

```
                               _______
                              |       |
arya-------------|       |
                              |       |
               _____          |       |
              |     |         | Switch |
tyrion--| Hub |--|       |
              |_____|         |       |
                 |            |_______|
cersei- ---*
```

- Can only sniff traffic to and from `tyrion`, but often that's enough
- If you have an old hub, keep it!
- Could also replace the switch by a hub, but that causes all kind of problems (performance, need access to the switch, etc)

ARP Cache poisoning

- Before `arya` contacts `tyrion`, she will ask for `tyrion`'s MAC address
- Idea: `cersei` can simply answer with his MAC address
- `arya` will update her ARP cache entry for `tyrion`'s IP address with `cersei`'s MAC address
- `arya` will then (unknowingly) talk to `cersei` instead of `tyrion`
- This attack is called *ARP spoofing* or *ARP cache poisoning*

ARP Cache poisoning

- Before `arya` contacts `tyrion`, she will ask for `tyrion`'s MAC address
- Idea: `cersei` can simply answer with his MAC address
- `arya` will update her ARP cache entry for `tyrion`'s IP address with `cersei`'s MAC address
- `arya` will then (unknowingly) talk to `cersei` instead of `tyrion`
- This attack is called *ARP spoofing* or *ARP cache poisoning*
- It gets better than that:
  - `cersei` does not have to wait for `arya` to send an ARP request
  - *Gratuitous ARP* packets are announcements ("replies without a request")

ARP Cache poisoning

- Before `arya` contacts `tyrion`, she will ask for `tyrion`'s MAC address
- Idea: `cersei` can simply answer with his MAC address
- `arya` will update her ARP cache entry for `tyrion`'s IP address with `cersei`'s MAC address
- `arya` will then (unknowingly) talk to `cersei` instead of `tyrion`
- This attack is called *ARP spoofing* or *ARP cache poisoning*
- It gets better than that:
  - `cersei` does not have to wait for `arya` to send an ARP request
  - *Gratuitous ARP* packets are announcements ("replies without a request")
- Various good reasons for gratuitous ARP:
  - Announce IP+MAC at boot time
  - Announce changed IP address to other hosts
  - IP-address takeover in high-performance clusters

`arpspoof` – convenient ARP spoofing

- Let's become man in the middle on `cersei` (192.168.42.1) between 192.168.42.2 and 192.168.42.3

`arpspoof` – convenient ARP spoofing

- Let's become man in the middle on `cersei` (192.168.42.1) between 192.168.42.2 and 192.168.42.3
- First enable IP forwarding in the Linux kernel:

```
root@cersei# echo 1 > /proc/sys/net/ipv4/ip_forward
```

arpspoof – convenient ARP spoofing

- Let's become man in the middle on `cersei` (192.168.42.1) between 192.168.42.2 and 192.168.42.3
- First enable IP forwarding in the Linux kernel:

```
root@cersei# echo 1 > /proc/sys/net/ipv4/ip_forward
```

- Now poison the ARP cache of 192.168.42.2:

```
root@cersei# arpspoof -t 192.168.42.2 192.168.42.3
```

arpspoof – convenient ARP spoofing

- Let's become man in the middle on `cersei` (192.168.42.1) between 192.168.42.2 and 192.168.42.3
- First enable IP forwarding in the Linux kernel:

```
root@cersei# echo 1 > /proc/sys/net/ipv4/ip_forward
```

- Now poison the ARP cache of 192.168.42.2:

```
root@cersei# arpspoof -t 192.168.42.2 192.168.42.3
```

- ... and the ARP cache of 192.168.42.3:

```
root@cersei# arpspoof -t 192.168.42.3 192.168.42.2
```

arpspoof – convenient ARP spoofing

- Let's become man in the middle on `cersei` (192.168.42.1) between 192.168.42.2 and 192.168.42.3
- First enable IP forwarding in the Linux kernel:

```
root@cersei# echo 1 > /proc/sys/net/ipv4/ip_forward
```

- Now poison the ARP cache of 192.168.42.2:

```
root@cersei# arpspoof -t 192.168.42.2 192.168.42.3
```

- ... and the ARP cache of 192.168.42.3:

```
root@cersei# arpspoof -t 192.168.42.3 192.168.42.2
```

- Now use your favorite sniffer on `cersei` to see traffic between 192.168.42.2 and 192.168.42.3
- Remark: `arpspoof` is part of the dsniff suite

Ettercap

- Very versatile tool for various low-level (ARP related) network attacks: ettercap
- Text mode and different GUIs
- Some features of Ettercap:
  - ARP cache poisoning
  - MAC flooding
  - Injection attacks
  - Support for plugins
  - OS fingerprinting

Ettercap

- Very versatile tool for various low-level (ARP related) network attacks: ettercap
- Text mode and different GUIs
- Some features of Ettercap:
  - ARP cache poisoning
  - MAC flooding
  - Injection attacks
  - Support for plugins
  - OS fingerprinting
- Very simple (but important) aspect: find all hosts of the network
- Simply send ARP requests for all hosts on the (sub)-network

RFCs

- RFC stands for *Request for Comments*
- Publications by the Internet Engineering Taskforce (IETF)
- Typically describes methods to operate the Internet
- Some RFCs are adopted as standards
- Published in plain ASCII
- Find them at `http://tools.ietf.org/rfc/rfcXXXX.txt`, where XXXX is the RFC number

RFCs

- RFC stands for *Request for Comments*
- Publications by the Internet Engineering Taskforce (IETF)
- Typically describes methods to operate the Internet
- Some RFCs are adopted as standards
- Published in plain ASCII
- Find them at `http://tools.ietf.org/rfc/rfcXXXX.txt`, where XXXX is the RFC number
- Not all RFCs are entirely serious, e.g., RFC 1149 "A Standard for the Transmission of IP Datagrams on Avian Carriers"

RFCs

- RFC stands for *Request for Comments*
- Publications by the Internet Engineering Taskforce (IETF)
- Typically describes methods to operate the Internet
- Some RFCs are adopted as standards
- Published in plain ASCII
- Find them at `http://tools.ietf.org/rfc/rfcXXXX.txt`, where XXXX is the RFC number
- Not all RFCs are entirely serious, e.g., RFC 1149 "A Standard for the Transmission of IP Datagrams on Avian Carriers"
- This RFC has actually been implemented, see `https://www.blug.linux.no/rfc1149/writeup/`

From RFC 826 – An Ethernet Address Resolution Protocol

```
?Do I have the hardware type in ar$hrd?
Yes: (almost definitely)
  [optionally check the hardware length ar$hln]
  ?Do I speak the protocol in ar$pro?
  Yes:
    [optionally check the protocol length ar$pln]
    Merge_flag := false
    If the pair <protocol type, sender protocol address> is
        already in my translation table, update the sender
        hardware address field of the entry with the new
        information in the packet and set Merge_flag to true.
    ?Am I the target protocol address?
    Yes:
      If Merge_flag is false, add the triplet <protocol type,
          sender protocol address, sender hardware address> to
          the translation table.
      ?Is the opcode ares_op$REQUEST?  (NOW look at the opcode!!)
      Yes:
        Swap hardware and protocol fields, putting the local
            hardware and protocol addresses in the sender fields.
        Set the ar$op field to ares_op$REPLY
        Send the packet to the (new) target hardware address on
            the same hardware on which the request was received.
```

Another kind of ARP spoofing

- First a clarification of terms:
  - `ar$hrd`: Hardware address space, e.g., Ethernet
  - `ar$pro`: Protocol address space
  - `ar$hln`: byte length of each hardware address
  - `ar$pln`: byte length of each protocol address
  - `ares_op$REQUEST/REPLY`: Distinguish type (opcode)

Another kind of ARP spoofing

- First a clarification of terms:
  - `ar$hrd`: Hardware address space, e.g., Ethernet
  - `ar$pro`: Protocol address space
  - `ar$hln`: byte length of each hardware address
  - `ar$pln`: byte length of each protocol address
  - `ares_op$REQUEST/REPLY`: Distinguish type (opcode)
- Note: Update the table entry *before* checking REQUEST/REPLY
- Does not depend on gratuitous ARP
- Can use ARP **requests** to poison the ARP cache

Another kind of ARP spoofing

- First a clarification of terms:
  - `ar$hrd`: Hardware address space, e.g., Ethernet
  - `ar$pro`: Protocol address space
  - `ar$hln`: byte length of each hardware address
  - `ar$pln`: byte length of each protocol address
  - `ares_op$REQUEST/REPLY`: Distinguish type (opcode)
- Note: Update the table entry *before* checking REQUEST/REPLY
- Does not depend on gratuitous ARP
- Can use ARP **requests** to poison the ARP cache
- Can also create new entries in the ARP cache, not just overwrite existing ones

Another kind of ARP spoofing

- First a clarification of terms:
  - `ar$hrd`: Hardware address space, e.g., Ethernet
  - `ar$pro`: Protocol address space
  - `ar$hln`: byte length of each hardware address
  - `ar$pln`: byte length of each protocol address
  - `ares_op$REQUEST/REPLY`: Distinguish type (opcode)
- Note: Update the table entry *before* checking REQUEST/REPLY
- Does not depend on gratuitous ARP
- Can use ARP **requests** to poison the ARP cache
- Can also create new entries in the ARP cache, not just overwrite existing ones
- Ettercap supports this: set `arp_poison_request = 1` in `/etc/ettercap/etter.conf`

MAC flooding

- A switch stores MAC-address-port mappings in a Content addressable memory (CAM) table
- Attack idea: send many Ethernet frames with different source MAC addresses
- Overflow the table

MAC flooding

- A switch stores MAC-address-port mappings in a Content addressable memory (CAM) table
- Attack idea: send many Ethernet frames with different source MAC addresses
- Overflow the table
- Effects of this depend highly on the switch
- Some (many?) switches will fall back to behave like a hub

ARP-attack countermeasures

- For small networks: static ARP table entries

ARP-attack countermeasures

- For small networks: static ARP table entries
- Disable gratuitous ARP (this may break things)
- Also, it's not so easy (at least on Linux):
  - `echo 0 > /proc/sys/net/ipv4/conf/eth0/arp_accept` is sometimes claimed to disable gratuitous ARP
  - Disables gratuitous ARP only for *new* IP addresses!

ARP-attack countermeasures

- For small networks: static ARP table entries
- Disable gratuitous ARP (this may break things)
- Also, it's not so easy (at least on Linux):
  - `echo 0 > /proc/sys/net/ipv4/conf/eth0/arp_accept` is sometimes claimed to disable gratuitous ARP
  - Disables gratuitous ARP only for *new* IP addresses!
- Fine-grained configuration through ARP filter `arptables`

ARP-attack countermeasures

- For small networks: static ARP table entries
- Disable gratuitous ARP (this may break things)
- Also, it's not so easy (at least on Linux):
  - `echo 0 > /proc/sys/net/ipv4/conf/eth0/arp_accept` is sometimes claimed to disable gratuitous ARP
  - Disables gratuitous ARP only for *new* IP addresses!
- Fine-grained configuration through ARP filter `arptables`
- Use `arpwatch` to monitor ARP messages

ARP-attack countermeasures

- For small networks: static ARP table entries
- Disable gratuitous ARP (this may break things)
- Also, it's not so easy (at least on Linux):
    - `echo 0 > /proc/sys/net/ipv4/conf/eth0/arp_accept` is sometimes claimed to disable gratuitous ARP
    - Disables gratuitous ARP only for *new* IP addresses!
- Fine-grained configuration through ARP filter `arptables`
- Use `arpwatch` to monitor ARP messages
- Protection mechanisms on advanced switches (like "Dynamic ARP Inspection" on Cisco switches)

ARP-attack countermeasures

- For small networks: static ARP table entries
- Disable gratuitous ARP (this may break things)
- Also, it's not so easy (at least on Linux):
  - `echo 0 > /proc/sys/net/ipv4/conf/eth0/arp_accept` is sometimes claimed to disable gratuitous ARP
  - Disables gratuitous ARP only for *new* IP addresses!
- Fine-grained configuration through ARP filter `arptables`
- Use `arpwatch` to monitor ARP messages
- Protection mechanisms on advanced switches (like "Dynamic ARP Inspection" on Cisco switches)
- Generally it's hard to defend against ARP spoofing, because

  ARP does not have any authentication mechanism

VLANs

- Advanced switches support partitioning of a local-area network (LAN) into multiple *virtual LANs* (VLANs)
- You can think of a VLAN as physically separated LANs (but easier to manage)
  - VLANs are separate broadcast domains
  - ARP requests/replies don't go from one VLAN to another

VLANs

- Advanced switches support partitioning of a local-area network (LAN) into multiple *virtual LANs* (VLANs)
- You can think of a VLAN as physically separated LANs (but easier to manage)
  - VLANs are separate broadcast domains
  - ARP requests/replies don't go from one VLAN to another
- Example use:
  - Logical network for staff of university institute
  - Logical network containing computers with student access

VLANs

- Advanced switches support partitioning of a local-area network (LAN) into multiple *virtual LANs* (VLANs)
- You can think of a VLAN as physically separated LANs (but easier to manage)
  - VLANs are separate broadcast domains
  - ARP requests/replies don't go from one VLAN to another
- Example use:
  - Logical network for staff of university institute
  - Logical network containing computers with student access
- This does not prevent ARP-level attacks
- Can limit the damage caused by ARP-level attacks ("students can only attack each other")

MAC address filtering

- Switches are aware of MAC addresses
- Idea: Switches can just ignore computers with unknown MAC addresses
- Set up a *white list* of MAC addresses on the switch

MAC address filtering

- Switches are aware of MAC addresses
- Idea: Switches can just ignore computers with unknown MAC addresses
- Set up a *white list* of MAC addresses on the switch
- Question: Can `cersei` spoof her MAC address?

MAC address filtering

- Switches are aware of MAC addresses
- Idea: Switches can just ignore computers with unknown MAC addresses
- Set up a *white list* of MAC addresses on the switch
- Question: Can `cersei` spoof her MAC address?

> *"A MAC address is a unique character string, and since it identifies a specific physical device – one individual NIC – the MAC address, by convention, never changes for the life of the NIC. [...] Because your NIC's MAC address is permanent, it's often referred to as the "real," or physical, address of a computer."*

http://www.watchguard.com/infocenter/editorial/135250.asp

MAC spoofing

> *"It is possible to spoof the MAC address, so an attacker could potentially capture details about a MAC address from your network and pretend to be that device to connect to your network, but no casual hacker or curious snooper will go to those lengths so MAC filtering will still protect you from the majority of users."*

(http://netsecurity.about.com/od/quicktip1/qt/qtwifimacfilter.htm)

MAC spoofing

> *"It is possible to spoof the MAC address, so an attacker could potentially capture details about a MAC address from your network and pretend to be that device to connect to your network, but no casual hacker or curious snooper will go to those lengths so MAC filtering will still protect you from the majority of users."*

(http://netsecurity.about.com/od/quicktip1/qt/qtwifimacfilter.htm)

- Going to "those lengths" means the following:

```
root@cersei# ip link set dev eth0 down
root@cersei# ip link set dev eth0 address 42:42:42:42:42:42
root@cersei# ip link set dev eth0 up
```

- Obviously, `42:42:42:42:42:42` can be any MAC address

More applications of MAC spoofing

- It is widely believed that MAC spoofing is hard
- Various “security” concepts rely on the fact that you cannot spoof MAC addresses:

More applications of MAC spoofing

- It is widely believed that MAC spoofing is hard
- Various "security" concepts rely on the fact that you cannot spoof MAC addresses:
  - Some proprietary software uses MAC addresses to bind a copy of the software to a fixed computer

More applications of MAC spoofing

- It is widely believed that MAC spoofing is hard
- Various "security" concepts rely on the fact that you cannot spoof MAC addresses:
  - Some proprietary software uses MAC addresses to bind a copy of the software to a fixed computer
  - In some public places each MAC address gets free WiFi for a fixed amount of time

More applications of MAC spoofing

- It is widely believed that MAC spoofing is hard
- Various “security” concepts rely on the fact that you cannot spoof MAC addresses:
  - Some proprietary software uses MAC addresses to bind a copy of the software to a fixed computer
  - In some public places each MAC address gets free WiFi for a fixed amount of time
- Summary: MAC spoofing is *easy*
- Security based on MAC uniqueness is bad

More applications of MAC spoofing

- It is widely believed that MAC spoofing is hard
- Various "security" concepts rely on the fact that you cannot spoof MAC addresses:
  - Some proprietary software uses MAC addresses to bind a copy of the software to a fixed computer
  - In some public places each MAC address gets free WiFi for a fixed amount of time
- Summary: MAC spoofing is *easy*
- Security based on MAC uniqueness is bad
- Better idea: Use IEEE 802.1X for port-based network access control (PNAC)

Wireless Networks

- Most important standard for wireless networks: IEEE 802.11 (released 1997)
- Designed to behave in many ways like a wired network
- Uses the same kind of MAC addresses as Ethernet
- *Association* with a network corresponds to plugging in the network cable

Wireless Networks

- Most important standard for wireless networks: IEEE 802.11 (released 1997)
- Designed to behave in many ways like a wired network
- Uses the same kind of MAC addresses as Ethernet
- *Association* with a network corresponds to plugging in the network cable
- Networks in vicinity are logically separated by their *network names* (service set identification, SSID)
- Communication is physically separated by using different channels (frequencies)

Wireless Networks

- Most important standard for wireless networks: IEEE 802.11 (released 1997)
- Designed to behave in many ways like a wired network
- Uses the same kind of MAC addresses as Ethernet
- *Association* with a network corresponds to plugging in the network cable
- Networks in vicinity are logically separated by their *network names* (service set identification, SSID)
- Communication is physically separated by using different channels (frequencies)
- Two different modes of operation:
  - Ad-hoc mode: peer-to-peer communication between nodes
  - Infrastructure mode: communication through *access point* (AP)
- Typical (and recommended) setup for permanent installations: infrastructure mode (managed mode)

Connecting to a WiFi network

- Connections to a wireless network are handled through management frames
- APs send beacon frames (by default, 10/second) containing:
  - Timestamp
  - Beacon interval
  - SSID
  - Frequency-hopping parameters

Connecting to a WiFi network

- Connections to a wireless network are handled through management frames
- APs send beacon frames (by default, 10/second) containing:
  - Timestamp
  - Beacon interval
  - SSID
  - Frequency-hopping parameters
- Connecting to a network:
  - Client sends authentication request with its identification (MAC), the network's SSID, etc.
  - If AP decides to accept the client, sends authentication OK
  - Client sends association request
  - AP sends association OK

Connecting to a WiFi network

- Connections to a wireless network are handled through management frames
- APs send beacon frames (by default, 10/second) containing:
  - Timestamp
  - Beacon interval
  - SSID
  - Frequency-hopping parameters
- Connecting to a network:
  - Client sends authentication request with its identification (MAC), the network's SSID, etc.
  - If AP decides to accept the client, sends authentication OK
  - Client sends association request
  - AP sends association OK
- Other important management frames
  - Reassociate request/response frames: change the AP
  - Disassociate frame: leave the network

Hidden SSID

- Clients need to know the SSID to authenticate/associate
- Idea: Don't send this SSID in beacon frames (legitimate users know it anyway)
- Advertise this as "network cloaking"

Hidden SSID

- Clients need to know the SSID to authenticate/associate
- Idea: Don't send this SSID in beacon frames (legitimate users know it anyway)
- Advertise this as "network cloaking"
- Network is not really cloaked, SSID is contained also in other frames
- Clearly *security by obscurity*, bad practice

Hidden SSID

- Clients need to know the SSID to authenticate/associate
- Idea: Don't send this SSID in beacon frames (legitimate users know it anyway)
- Advertise this as "network cloaking"
- Network is not really cloaked, SSID is contained also in other frames
- Clearly *security by obscurity*, bad practice
- Not intended by the standard
- Windows XP machines always prefer access points that broadcast their SSID
- Very easy to lure those machines into a fake AP

Nice summary on hidden SSIDs

> *"Do you ever wonder sometimes how it is that some ideas just won't die? Like the thought that not broadcasting your wireless network's SSID will somehow make you more secure? This is a myth that needs to be forcibly dragged out behind the woodshed, strangled until it wheezes its last labored breath, then shot several times for good measure."* —*Steve Riley*

http://blogs.technet.com/b/steriley/archive/2007/10/16/myth-vs-reality-wireless-ssids.aspx

Encrypted WiFi part I

- WiFi is even more susceptible to eavesdropping attacks than cable-based networks
- Idea of original 802.11: use encryption to provide "Wired Equivalent Privacy" (WEP)

Encrypted WiFi part I

- WiFi is even more susceptible to eavesdropping attacks than cable-based networks
- Idea of original 802.11: use encryption to provide "Wired Equivalent Privacy" (WEP)
- Encryption uses RC4 with 40-bit key and 24-bit IV
- Access to a wireless LAN only with pre-shared key
- Weak encryption because of US export laws at the time of standardization
- Once export restrictions got lifted, extend to 104-bit key (WEP-104)

Encrypted WiFi part I

- WiFi is even more susceptible to eavesdropping attacks than cable-based networks
- Idea of original 802.11: use encryption to provide "Wired Equivalent Privacy" (WEP)
- Encryption uses RC4 with 40-bit key and 24-bit IV
- Access to a wireless LAN only with pre-shared key
- Weak encryption because of US export laws at the time of standardization
- Once export restrictions got lifted, extend to 104-bit key (WEP-104)
- Two possible authentication mechanisms:
  - Open System Authentication: no authentication
  - Challenge-Response Authentication

Encrypted WiFi part I

- WiFi is even more susceptible to eavesdropping attacks than cable-based networks
- Idea of original 802.11: use encryption to provide "Wired Equivalent Privacy" (WEP)
- Encryption uses RC4 with 40-bit key and 24-bit IV
- Access to a wireless LAN only with pre-shared key
- Weak encryption because of US export laws at the time of standardization
- Once export restrictions got lifted, extend to 104-bit key (WEP-104)
- Two possible authentication mechanisms:
  - Open System Authentication: no authentication
  - Challenge-Response Authentication
- WEP was optional; in the early days of WiFi most networks were not encrypted

RC4

- Stream cipher designed by Rivest in 1987
- Algorithm was a trade secret of RSA Security
- Posted anonymously to the cypherpunks mailing list in 1994

RC4

- Stream cipher designed by Rivest in 1987
- Algorithm was a trade secret of RSA Security
- Posted anonymously to the cypherpunks mailing list in 1994
- Two parts of the algorithm: key schedule and pseudo-random generation
- Supports keys of length between 1 and 256 bytes
- Very small and simple C code, quickly became popular

RC4/WEP security

- One major problem: short 24-bit IVs
- After ≈ 5000 frames there is a 50% chance of repeating IVs

RC4/WEP security

- One major problem: short 24-bit IVs
- After ≈ 5000 frames there is a 50% chance of repeating IVs
- Another problem: WEP concatenates key and IV to input for key schedule
- Fluhrer, Mantin, Shamir, 2001: First few output bytes of RC4 are biased
- FMS attack against WEP uses biases to find the master key

RC4/WEP security

- One major problem: short 24-bit IVs
- After ≈ 5000 frames there is a 50% chance of repeating IVs
- Another problem: WEP concatenates key and IV to input for key schedule
- Fluhrer, Mantin, Shamir, 2001: First few output bytes of RC4 are biased
- FMS attack against WEP uses biases to find the master key
- Klein 2005: even more correlations between key and output

RC4/WEP security

- One major problem: short 24-bit IVs
- After ≈ 5000 frames there is a 50% chance of repeating IVs
- Another problem: WEP concatenates key and IV to input for key schedule
- Fluhrer, Mantin, Shamir, 2001: First few output bytes of RC4 are biased
- FMS attack against WEP uses biases to find the master key
- Klein 2005: even more correlations between key and output
- Tews, Weinmann, Pyshkin, 2005: tune Klein's attack for WEP-104
  - Break WEP key after 40000 frames with probability $> 50\%$
  - Break WEP key after 85000 frames with probability $> 95\%$

RC4/WEP security

- One major problem: short 24-bit IVs
- After ≈ 5000 frames there is a 50% chance of repeating IVs
- Another problem: WEP concatenates key and IV to input for key schedule
- Fluhrer, Mantin, Shamir, 2001: First few output bytes of RC4 are biased
- FMS attack against WEP uses biases to find the master key
- Klein 2005: even more correlations between key and output
- Tews, Weinmann, Pyshkin, 2005: tune Klein's attack for WEP-104
  - Break WEP key after 40000 frames with probability $> 50\%$
  - Break WEP key after 85000 frames with probability $> 95\%$
- Consequence: Can break WEP-104 in < 1 minute

aircrack-ng

- aircrack-ng is a suite of tools to break WiFi encryption
- Part of most Linux distributions

aircrack-ng

- aircrack-ng is a suite of tools to break WiFi encryption
- Part of most Linux distributions
- airmon-ng: switch the wireless interface into monitor mode, create monitor interface mon0

aircrack-ng

- aircrack-ng is a suite of tools to break WiFi encryption
- Part of most Linux distributions
- airmon-ng: switch the wireless interface into monitor mode, create monitor interface mon0
- airodump-ng: scan for networks, sniff and dump received data to a file
- aireplay-ng: generate traffic by replaying frames
- Can also be used, e.g., to de-authenticate/disassociate clients
  - Take the MAC address of that client if MAC filtering is enabled
  - Record the re-authentication of that client to obtain (hidden) SSID

aircrack-ng

- aircrack-ng is a suite of tools to break WiFi encryption
- Part of most Linux distributions
- airmon-ng: switch the wireless interface into monitor mode, create monitor interface mon0
- airodump-ng: scan for networks, sniff and dump received data to a file
- aireplay-ng: generate traffic by replaying frames
- Can also be used, e.g., to de-authenticate/disassociate clients
  - Take the MAC address of that client if MAC filtering is enabled
  - Record the re-authentication of that client to obtain (hidden) SSID
- aircrack: Use data recorded by airodump-ng to break encryption

aircrack-ng

- `aircrack-ng` is a suite of tools to break WiFi encryption
- Part of most Linux distributions
- `airmon-ng`: switch the wireless interface into monitor mode, create monitor interface `mon0`
- `airodump-ng`: scan for networks, sniff and dump received data to a file
- `aireplay-ng`: generate traffic by replaying frames
- Can also be used, e.g., to de-authenticate/disassociate clients
  - Take the MAC address of that client if MAC filtering is enabled
  - Record the re-authentication of that client to obtain (hidden) SSID
- `aircrack`: Use data recorded by `airodump-ng` to break encryption
- The `aircrack` suite has various filters and switches that influence performance
- Generally a very powerful tool to break WiFi encryption (see homework)

Encrypted WiFi part II

- Drop-in replacement for WEP: WiFi Protected Access (WPA)
- Available since 2003 (draft of 802.11i)
- Uses Temporal Key Integrity Protocol (TKIP) for encryption
- TKIP is a wrapper around RC4 to address some weaknesses in WEP

Encrypted WiFi part II

- Drop-in replacement for WEP: WiFi Protected Access (WPA)
- Available since 2003 (draft of 802.11i)
- Uses Temporal Key Integrity Protocol (TKIP) for encryption
- TKIP is a wrapper around RC4 to address some weaknesses in WEP
- Changes compared to WEP:
  - Use 128-bit keys instead of 40-bit keys

Encrypted WiFi part II

- Drop-in replacement for WEP: WiFi Protected Access (WPA)
- Available since 2003 (draft of 802.11i)
- Uses Temporal Key Integrity Protocol (TKIP) for encryption
- TKIP is a wrapper around RC4 to address some weaknesses in WEP
- Changes compared to WEP:
  - Use 128-bit keys instead of 40-bit keys
  - Use 48-bit IV instead of 24-bit IV
  - Use a mixing function to combine key and IV

Encrypted WiFi part II

- Drop-in replacement for WEP: WiFi Protected Access (WPA)
- Available since 2003 (draft of 802.11i)
- Uses Temporal Key Integrity Protocol (TKIP) for encryption
- TKIP is a wrapper around RC4 to address some weaknesses in WEP
- Changes compared to WEP:
  - Use 128-bit keys instead of 40-bit keys
  - Use 48-bit IV instead of 24-bit IV
  - Use a mixing function to combine key and IV
  - Use IV as a sequence counter to protect against replay attacks (drop out-of-order packets)

Encrypted WiFi part II

- Drop-in replacement for WEP: WiFi Protected Access (WPA)
- Available since 2003 (draft of 802.11i)
- Uses Temporal Key Integrity Protocol (TKIP) for encryption
- TKIP is a wrapper around RC4 to address some weaknesses in WEP
- Changes compared to WEP:
  - Use 128-bit keys instead of 40-bit keys
  - Use 48-bit IV instead of 24-bit IV
  - Use a mixing function to combine key and IV
  - Use IV as a sequence counter to protect against replay attacks (drop out-of-order packets)
  - Use a rekeying mechanism to change the key every 10000 packets

Encrypted WiFi part II

- Drop-in replacement for WEP: WiFi Protected Access (WPA)
- Available since 2003 (draft of 802.11i)
- Uses Temporal Key Integrity Protocol (TKIP) for encryption
- TKIP is a wrapper around RC4 to address some weaknesses in WEP
- Changes compared to WEP:
  - Use 128-bit keys instead of 40-bit keys
  - Use 48-bit IV instead of 24-bit IV
  - Use a mixing function to combine key and IV
  - Use IV as a sequence counter to protect against replay attacks (drop out-of-order packets)
  - Use a rekeying mechanism to change the key every 10000 packets
- Reason for WPA/TKIP: No need to update hardware, "only" firmware

Weaknesses in WPA

- Core problem of WPA/TKIP: It still uses RC4

Weaknesses in WPA

- Core problem of WPA/TKIP: It still uses RC4
- AlFardan, Bernstein, Paterson, Poettering, Schuldt, 2013:
  Plaintext-recovery attack against RC4 in TLS

Weaknesses in WPA

- Core problem of WPA/TKIP: It still uses RC4
- AlFardan, Bernstein, Paterson, Poettering, Schuldt, 2013: Plaintext-recovery attack against RC4 in TLS
- Paterson, Poettering, Schuldt, 2014: Plaintext-recovery attack against RC4 in WPA/TKIP

Weaknesses in WPA

- Core problem of WPA/TKIP: It still uses RC4
- AlFardan, Bernstein, Paterson, Poettering, Schuldt, 2013: Plaintext-recovery attack against RC4 in TLS
- Paterson, Poettering, Schuldt, 2014: Plaintext-recovery attack against RC4 in WPA/TKIP
- Better attacks against RC4 by Garman, Paterson, and van der Merwe in 2015

Weaknesses in WPA

- Core problem of WPA/TKIP: It still uses RC4
- AlFardan, Bernstein, Paterson, Poettering, Schuldt, 2013: Plaintext-recovery attack against RC4 in TLS
- Paterson, Poettering, Schuldt, 2014: Plaintext-recovery attack against RC4 in WPA/TKIP
- Better attacks against RC4 by Garman, Paterson, and van der Merwe in 2015
- . . . and by Vanhoef and Piessens in 2015

Weaknesses in WPA

- Core problem of WPA/TKIP: It still uses RC4
- AlFardan, Bernstein, Paterson, Poettering, Schuldt, 2013: Plaintext-recovery attack against RC4 in TLS
- Paterson, Poettering, Schuldt, 2014: Plaintext-recovery attack against RC4 in WPA/TKIP
- Better attacks against RC4 by Garman, Paterson, and van der Merwe in 2015
- ... and by Vanhoef and Piessens in 2015
- ... and by Bricout, Murphy, Paterson, and van der Merwe in 2016

Encrypted WiFi part III

- In 2004, IEEE announces 802.11i (WPA2)
- Most important chance compared to WEP and WPA: get rid of RC4
- Use CCMP instead (AES in Counter Mode Cipher Block Chaining Message Authentication Code Protocol)
- 128-bit AES key

Encrypted WiFi part III

- In 2004, IEEE announces 802.11i (WPA2)
- Most important chance compared to WEP and WPA: get rid of RC4
- Use CCMP instead (AES in Counter Mode Cipher Block Chaining Message Authentication Code Protocol)
- 128-bit AES key
- Since 2006 all WiFi-certified cards need to support WPA2

Encrypted WiFi part III

- In 2004, IEEE announces 802.11i (WPA2)
- Most important chance compared to WEP and WPA: get rid of RC4
- Use CCMP instead (AES in Counter Mode Cipher Block Chaining Message Authentication Code Protocol)
- 128-bit AES key
- Since 2006 all WiFi-certified cards need to support WPA2
- Different ways of handling authentication and keys, easiest one: pre-shared-key (PSK)
- PSK is typically derived from a passphrase through a key-derivation-function (specifically, PBKDF2)

Everything fine with WPA2?

- Essentially three problems with WPA2:
  - Weak passphrases (`aircrack-ng` has support for brute force):

    "A key generated from a pass-phrase of less than about 20 characters is unlikely to deter attacks." —*IEEE 802.11i standard*

Everything fine with WPA2?

- Essentially three problems with WPA2:
  - Weak passphrases (`aircrack-ng` has support for brute force):

    "A key generated from a pass-phrase of less than about 20 characters is unlikely to deter attacks." —*IEEE 802.11i standard*

  - WPA2 still supports TKIP (with RC4) for backwards compatibility

Everything fine with WPA2?

- Essentially three problems with WPA2:
  - Weak passphrases (`aircrack-ng` has support for brute force):

    "A key generated from a pass-phrase of less than about 20 characters is unlikely to deter attacks." —*IEEE 802.11i standard*

  - WPA2 still supports TKIP (with RC4) for backwards compatibility
  - Many WPA2 routers support *WiFi Protected Setup* (WPS)

Everything fine with WPA2?

- Essentially three problems with WPA2:
  - Weak passphrases (`aircrack-ng` has support for brute force):

    "A key generated from a pass-phrase of less than about 20 characters is unlikely to deter attacks." —*IEEE 802.11i standard*

  - WPA2 still supports TKIP (with RC4) for backwards compatibility
  - Many WPA2 routers support *WiFi Protected Setup* (WPS)

WPS attack by Viehböck, 2011

- WPS defines various ways to set up a secure home network
- One way: Enter the AP's WPS PIN into the new network client (your laptop)

Everything fine with WPA2?

- Essentially three problems with WPA2:
  - Weak passphrases (`aircrack-ng` has support for brute force):

    "A key generated from a pass-phrase of less than about 20 characters is unlikely to deter attacks." —*IEEE 802.11i standard*

  - WPA2 still supports TKIP (with RC4) for backwards compatibility
  - Many WPA2 routers support *WiFi Protected Setup* (WPS)

WPS attack by Viehböck, 2011

- WPS defines various ways to set up a secure home network
- One way: Enter the AP's WPS PIN into the new network client (your laptop)
- PIN has 8 digits, 10^8 possibilities
- Two halves of PIN are checked indendently; brute-force takes only $2 \cdot 10^4$

Everything fine with WPA2?

- Essentially three problems with WPA2:
  - Weak passphrases (`aircrack-ng` has support for brute force):

    "A key generated from a pass-phrase of less than about 20 characters is unlikely to deter attacks." —*IEEE 802.11i standard*

  - WPA2 still supports TKIP (with RC4) for backwards compatibility
  - Many WPA2 routers support *WiFi Protected Setup* (WPS)

WPS attack by Viehböck, 2011

- WPS defines various ways to set up a secure home network
- One way: Enter the AP's WPS PIN into the new network client (your laptop)
- PIN has 8 digits, 10^8 possibilities
- Two halves of PIN are checked indendently; brute-force takes only $2 \cdot 10^4$
- Last bit of the PIN is a parity bit, hence, only 11000 guesses
- Brute-forcing this PIN takes < 4 hours

The cryptographer's response

- It is very hard to prevent an attacker from sniffing your communication
- It is even harder to prevent an attacker from disrupting your communication
- System administrators can do something, even harder for users
- Most of the network is typically not under your control

The cryptographer's response

- It is very hard to prevent an attacker from sniffing your communication
- It is even harder to prevent an attacker from disrupting your communication
- System administrators can do something, even harder for users
- Most of the network is typically not under your control
- Go for the safe worst-case assumption:

  Everybody can read and modify everything you send over the network.

The cryptographer's response

- It is very hard to prevent an attacker from sniffing your communication
- It is even harder to prevent an attacker from disrupting your communication
- System administrators can do something, even harder for users
- Most of the network is typically not under your control
- Go for the safe worst-case assumption:

  Everybody can read and modify everything you send over the network.
- Solution for confidentiality and integrity: end-to-end encrypt/authenticate everything.