

CIS 4930: Secure IoT

Prof. Kaushal Kafle

Lecture 19

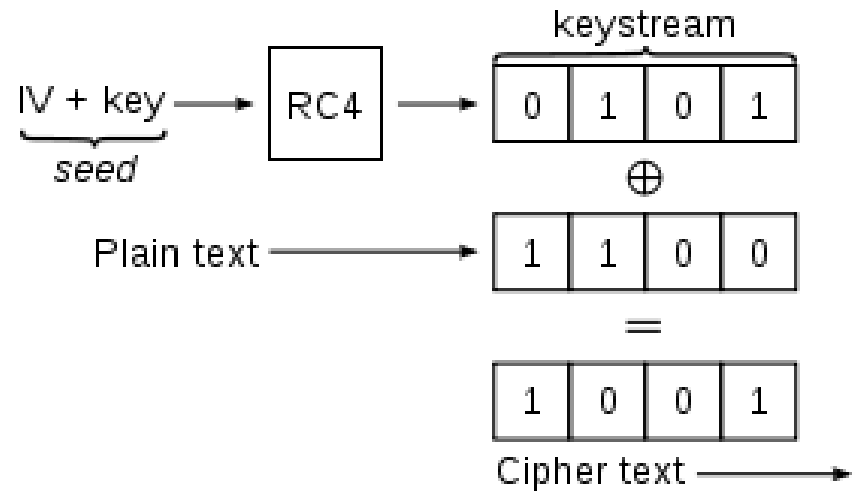
Wireless Security

Let's sprinkle on some of that
crypto magic sauce

Wired Equivalent Privacy (WEP)

- Part of original 802.11 standard
- Uses stream cipher:
 - Stream cipher crypto
 - $C = M \oplus S$, where S is pseudo-random sequence produced using a key K and an IV
 - $M = C \oplus S \rightarrow (M \oplus S) \oplus S = M$
- WEP uses RC4: supports seed up to 256 bits
 - seed = 24-bit IV + WEP key
- In WEPv1, key was 40 bits \rightarrow 64bit seed
- Later versions supported seeds of 128 and 256 bits

Wired Equivalent Privacy (WEP)



- Data transmission:
 - Produce keystream S using RC4 with seed function $f(K, IV)$
 - $C = M \oplus S$
 - send (IV, C) frames
 - knowledge of IV and K sufficient to decrypt C

WEP Authentication Modes

- **Open System:**
 - client doesn't need to provide any credentials
 - immediate association with access point
 - but can only send and receive info if using correct key
- **Shared Key:**
 - client must prove knowledge of WEP key before associating
 - AP sends client plaintext challenge; response is challenge encrypted with the correct key
- Q: Which is more secure?

WEP Shared Key Vulnerability

- Random Challenge: “jk4533hfdsa9”
- Response: {IV, “jk4533hfdsa9” \oplus RC4(K,IV)}
- here, RC4(K,IV) denotes RC4 encryption using a key derived from key K and IV
- Eavesdropper can observe plaintext challenge and encrypted response, and can produce:
 - challenge \oplus response = RC4(K,IV)
 - RC4(K,IV) sufficient to authenticate:
 - next challenge: “abcdef”
 - Eve responds (without knowing K!): {IV, “abcdef” \oplus RC4(K,IV)}

WEP Problems: IV Collisions

- IVs are too small... likely collision(s) after a few hours
- when IVs are the same, two ciphertexts can be xor'ed together to produce the xor of the plaintexts because it can generate the same OTP (*recall from previous classes!*)
- statistical analysis will then yield plaintexts
 - redundancy in IP packets makes this easy!
 - knowledge of protocols further limits the possibilities
 - or, attacker sends message thru Internet to a wireless client in a manner that will result a known response (e.g., ping message)
- if multiple messages share same IV, once one is recovered, others can be trivially/immediately recovered --**WHY?**

$$\rightarrow C1 = M1 \wedge (S+IV)$$

$$\rightarrow C2 = M2 \wedge (S+IV)$$

\rightarrow if M1 known, XOR C1 and C2 to recover M2

WEP Problems: Exploiting RC4 Weaknesses

- RC4 has a weakness: first few bytes of keystream are sometimes not particularly random looking [Fluhrer, Mantin and Shamir Attack; 2001]
- Mathematical result: Given enough keystreams, it's possible to construct the key [ciphertext-only attack]
- Attacker's goal: Get a lot of keystreams!
 - Basic approach: replay a bunch of ARP packets
 - AP will respond to replayed ARP
 - Sufficient number of AP's encrypted packets will yield key
- An aside: standard RC4 fix: discard first n bytes of keystream (usually $n \geq 3072$)

Story Time: TJX Data Breach



- TJX (TJMaxx + Marshalls + Bob's) main database compromised in 2007
 - ~45M credit and debit cards stolen over 18 months! <[Ars Technica news](#)>
- Scanning devices, cash registers, and PCs in Minnesota Marshalls wirelessly communicated to server, which communicated to backend database
- Wireless data encrypted using WEP
- WEP key stolen via wardriving attack from MN parking lot. Uh-oh.
- **Lesson: Don't use WEP!**

Wi-Fi Protected Access (WPA)

- Engineered to be the “secure replacement” for WEP
- Authentication stages:
 - Shared secret used to derive encryption keys
 - Client authenticates to AP
 - Encryption keys are used to produce keystreams for encrypting traffic

Wi-Fi Protected Access (WPA)

- Two Modes:
 - **PSK (Pre-shared Key):**
 - also called “WPA Personal”
 - shared secret manually entered into all devices
 - designed for home use
 - **802.1x Mode:**
 - also called “WPA Enterprise”
 - authentication handled by backend service (e.g., RADIUS server) via Extensible Authentication Protocol (EAP)
 - may make use of certificates or other authentication techniques
 - e.g., SaxonNet

Wi-Fi Protected Access (WPA)

- Encrypting Traffic (2 confidentiality protocols):
 - **Temporal Key Integrity Protocol (TKIP):**
 - uses RC4, but designed to improve upon WEP's shortcomings
 - increases size of IV to 48 bits
 - rather than just concatenate IV, uses more complex key mixing routine

Wi-Fi Protected Access (WPA)

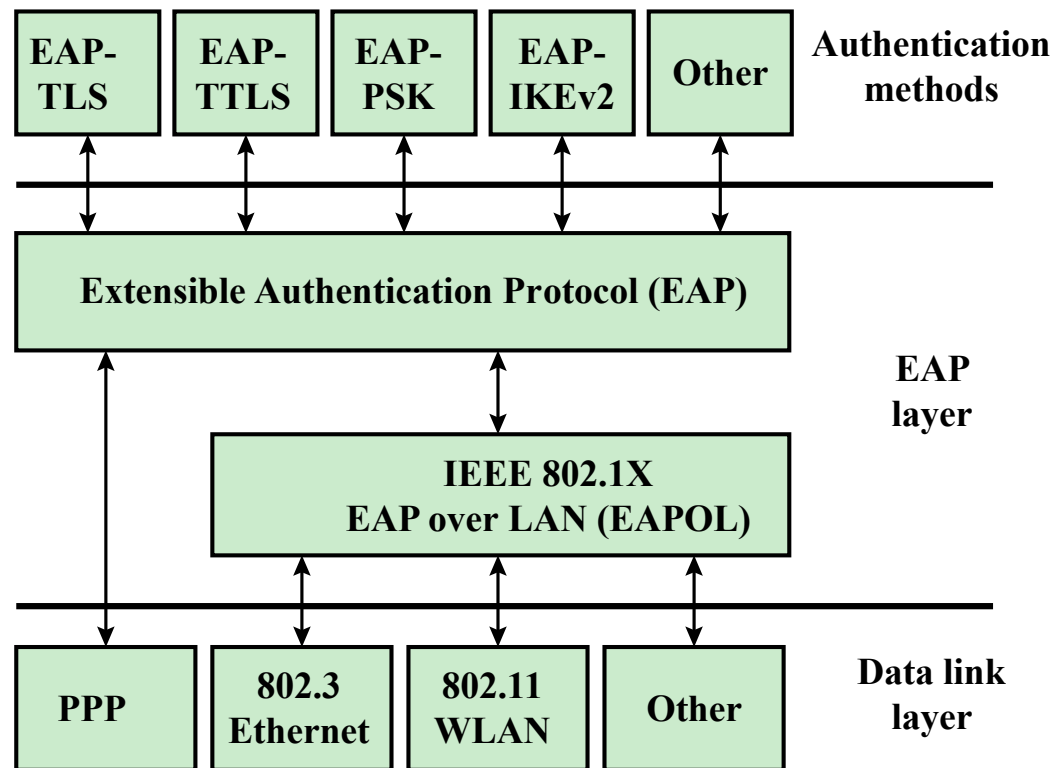
- Encrypting Traffic (2 confidentiality protocols):
 - **AES:**
 - supported in newer WPA2 protocol
 - runs AES in stream-cipher like way (e.g., using something similar to counter mode)
 - AES in CTR mode for stream-like behavior for data encryption (confidentiality)
 - AES in CBC-MAC mode for Integrity

Attacks against WPA

- WPA is a lot stronger than WEP
- Most attacks rely on weak passwords
- user-supplied keys are either entered as 256-bit string (64 hex digits) or as password
- password is hashed to produce key using 4096 iterations of HMAC-SHA1 with SSID of AP as salt
 - This makes brute-forcing computationally expensive!
- There exists dictionaries of pre-hashed keys for most popular SSIDs (“linksys”, “redsox”, “spectrum”, etc.)

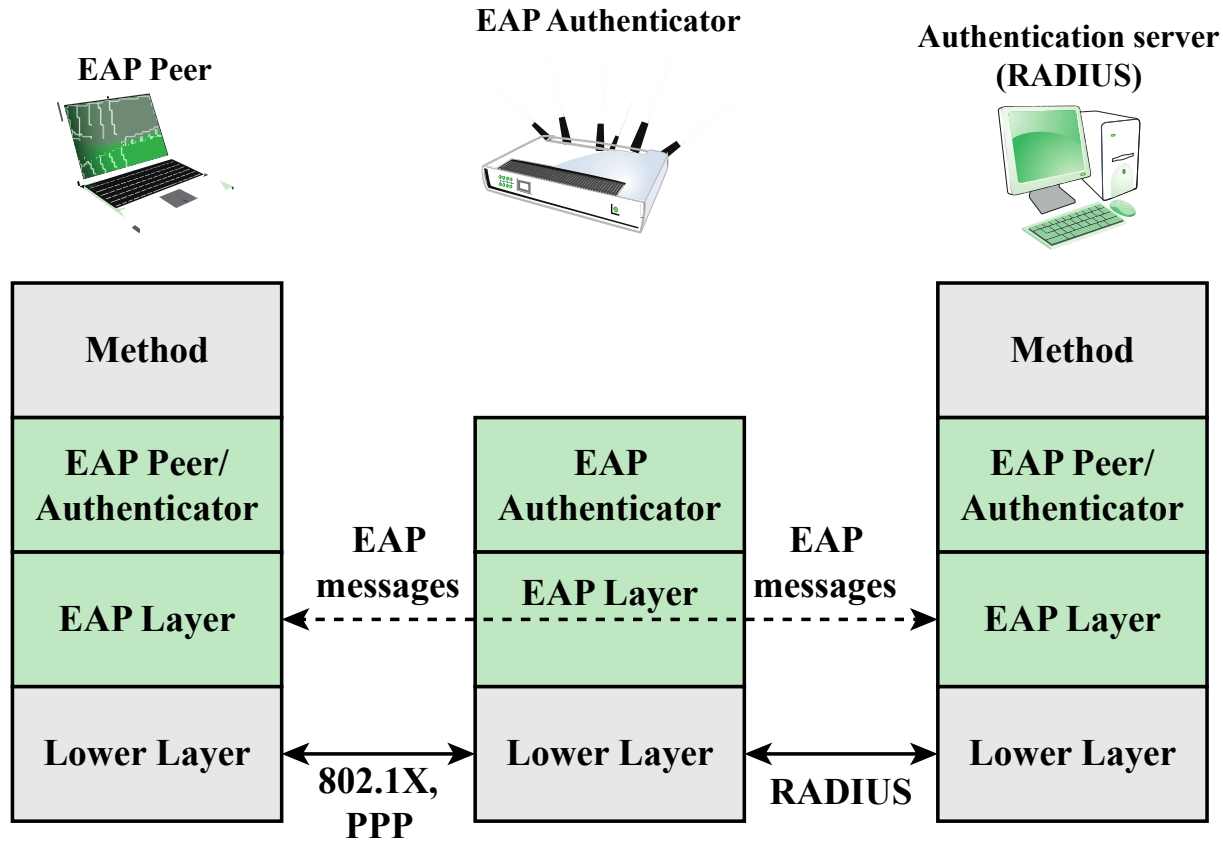
EAP Authentication

- EAP – Extensible Authentication Protocol
- WPA Enterprise using 802.1x for authentication



(from Stallings, Crypto and Net Security)

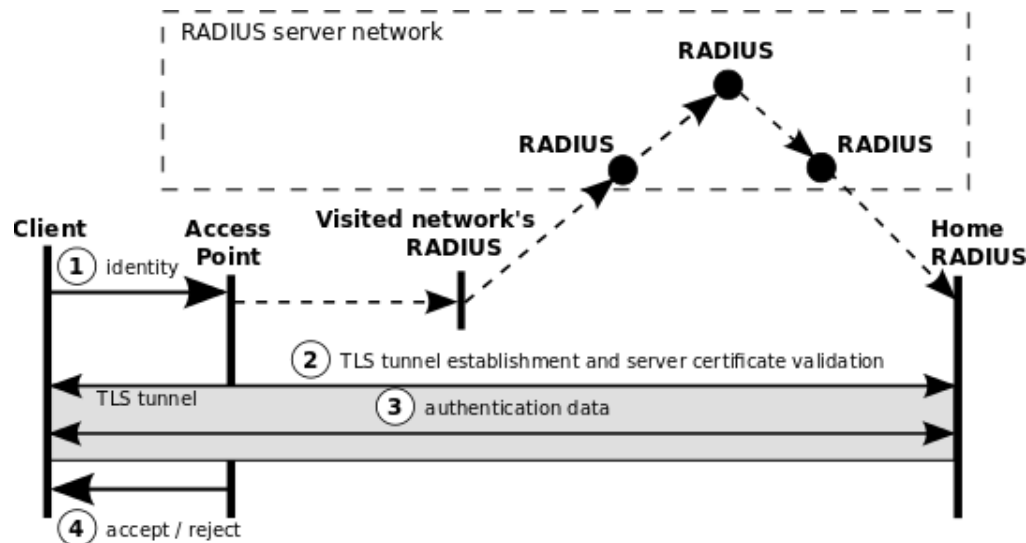
EAP Authentication



(from Stallings, Crypto and Net Security)

Eduroam

- Many campuses have started deploying Eduroam for WiFi (https://monitor.eduroam.org/map_service_loc.php)
 - Better than MAC address-based authentication (based on 802.1x protocol)
 - Allows you to access WiFi at other universities



(<https://dl.acm.org/doi/pdf/10.1145/2766498.2766512>)

Eduroam Config Woes

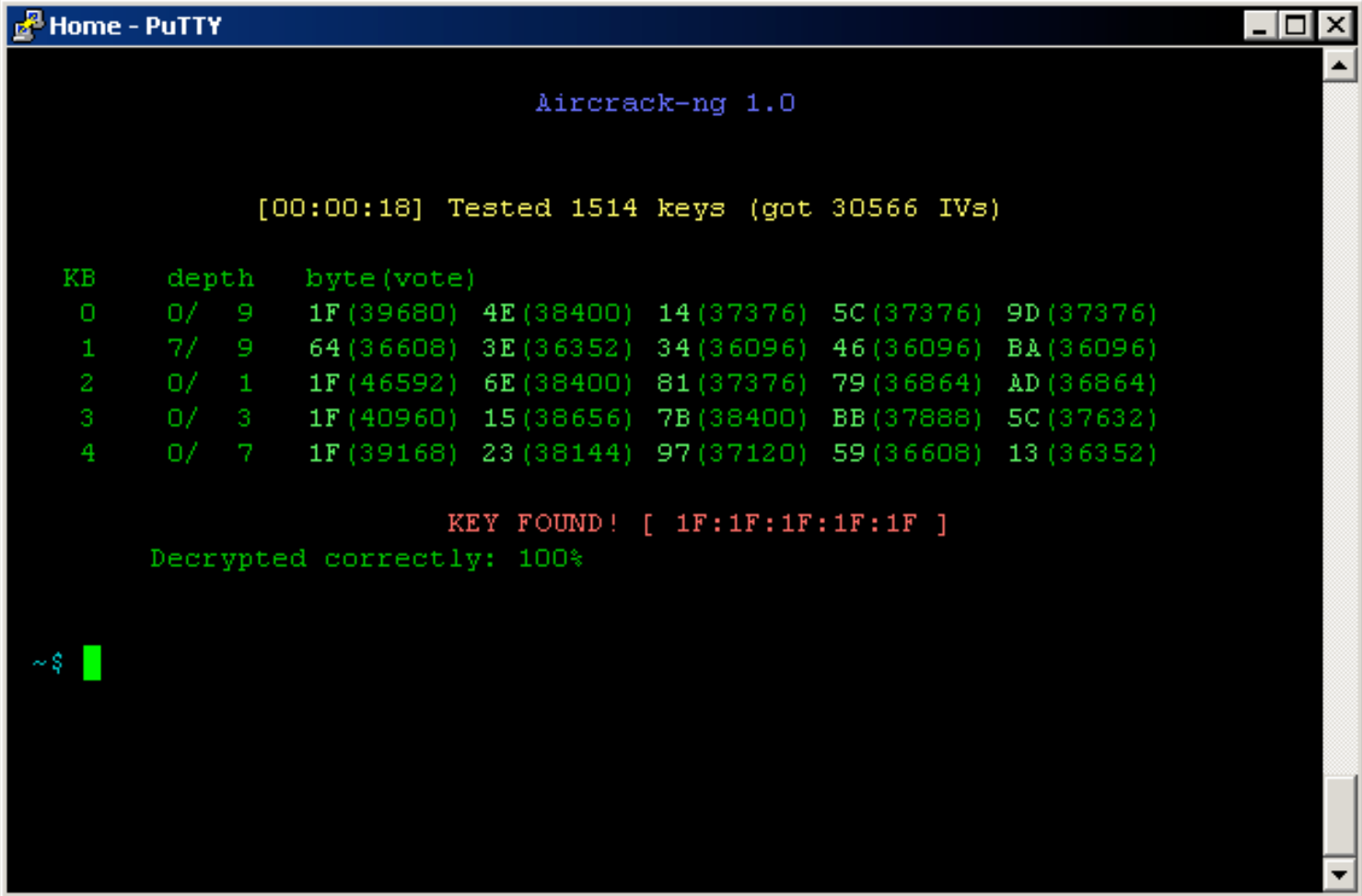
- Brenza et al. “A Practical Investigation of Identity Theft Vulnerabilities in Eduroam.” WiSec 2015.
- <https://dl.acm.org/doi/pdf/10.1145/2766498.2766512>
- Found that many Eduroam clients contain configuration errors
 - Root CA not set, so any certificate pushed by the AP will be valid. (Configure your root CA!)
 - Deficient certificate validation / Missing server name check / common name check
 - Often problem with the platform
- Enables **Evil Twin attack** (associating with rogue AP), which can result in
 - Eavesdropping or modifying traffic
 - Exposure of passwords (arguably more severe)
 - E.g., user’s device does not perform server’s certificate validation!

Password Exposure

- The most common authentication method in Eduroam uses TLS to create a secure tunnel for password authentication
 - *PEAP (Protected Extensible Authentication Protocol) or EAP-TTLS (Tunneled TLS)* for the tunnel
 - Password authentication via
 - EAP-PAP: cleartext password (*supposedly protected by the tunnel*)
 - EAP-MSCHAPv2: challenge response, but can still brute-force passwords offline by collecting challenge/response (known to be a weak protocol)
 - *However, we have misconfigured clients*
- Use EAP-TLS -> both server/client authentication using certificates
 - **Why does this help?**
 - MITM, but no threat of password exposure.

Practical Attacks

Plenty of tools available (usually exploit RC4 weakness)



```
Home - PuTTY
Aircrack-ng 1.0

[00:00:18] Tested 1514 keys (got 30566 IVs)

KB    depth  byte (vote)
0     0/ 9    1F (39680) 4E (38400) 14 (37376) 5C (37376) 9D (37376)
1     7/ 9    64 (36608) 3E (36352) 34 (36096) 46 (36096) BA (36096)
2     0/ 1    1F (46592) 6E (38400) 81 (37376) 79 (36864) AD (36864)
3     0/ 3    1F (40960) 15 (38656) 7B (38400) BB (37888) 5C (37632)
4     0/ 7    1F (39168) 23 (38144) 97 (37120) 59 (36608) 13 (36352)

KEY FOUND! [ 1F:1F:1F:1F:1F ]
Decrypted correctly: 100%

~$ █
```

MitM with airbase-ng

```
AIRBASE-NG(1) 1. sh AIRBASE-NG(1)

NAME
airbase-ng - multi-purpose tool aimed at attacking clients as opposed to the Access
Point (AP) itself

SYNOPSIS
airbase-ng [options] <interface name>

DESCRIPTION
airbase-ng is multi-purpose tool aimed at attacking clients as opposed to the Access
Point (AP) itself. Since it is so versatile and flexible, summarizing it is a chal-
lenge. Here are some of the feature highlights:
- Implements the Caffe Latte WEP client attack
- Implements the Hirte WEP client attack
- Ability to cause the WPA/WPA2 handshake to be captured
- Ability to act as an ad-hoc Access Point
- Ability to act as a full Access Point
- Ability to filter by SSID or client MAC addresses
- Ability to manipulate and resend packets
- Ability to encrypt sent packets and decrypt received packets

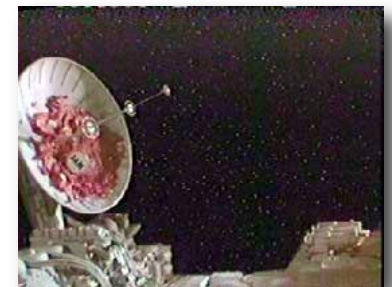
The main idea is of the implementation is that it should encourage clients to associate
with the fake AP, not prevent them from accessing the real AP.

A tap interface (atX) is created when airbase-ng is run. This can be used to receive
decrypted packets or to send encrypted packets.

As real clients will most probably send probe requests for common/configured networks,
```

Jamming

- Wireless signals are subject to jamming as they are radio waves
- **Analog Jamming:** Emit continuous garbage/random analog signal on the same frequency as the target
- **Standard defense:** spread spectrum
 - Spread the signal over wider frequency
- **Digital Jamming:** Capture digitally encoded data and corrupt it during transmission
 - E.g. bit flipping, injecting random data to mess with decoding
 - Potential defense: error correction, encryption



Filtering: Firewalls

- Filtering traffic based on **policy**
 - Policy determines what is acceptable traffic
 - *Access control* over traffic
 - Accept or deny
- May perform other duties
 - Logging (forensics, SLA)
 - Flagging (intrusion detection)
 - QoS (differentiated services e.g VoIP)



IP Firewall Policy

- Specifies what traffic is (not) allowed
 - Maps attributes to address and ports
 - Example: HTTP should be allowed to any external host, but inbound only to web-server
 - Rules typically refer to IP addresses, not hostnames -- **WHY?**

Source		Destination		Protocol	Flags	Actions
Address	Port	Address	Port			
*	*	1.1.1.1	80	TCP	SYN	Accept
1.1.1.*	*	*	80	TCP	SYN	Accept
*	*	*	80	TCP		Accept
*	*	*	*	TCP		Deny

Default accept vs. Default deny

- Default policy specifies what to do if no other policy applies
- Most OSes default to default accept
- Most organizations default to default deny

Source		Destination		Protocol	Flags	Actions
Address	Port	Address	Port			
*	*	1.1.1.1	80	TCP	SYN	Accept
1.1.1.*	*	*	80	TCP	SYN	Accept
*	*	*	80	TCP		Accept
*	*	*	*	TCP		Deny

- **Blacklisting**

- Specifies connectivity that is explicitly disallowed
 - E.g., prevent connections from badguys.com

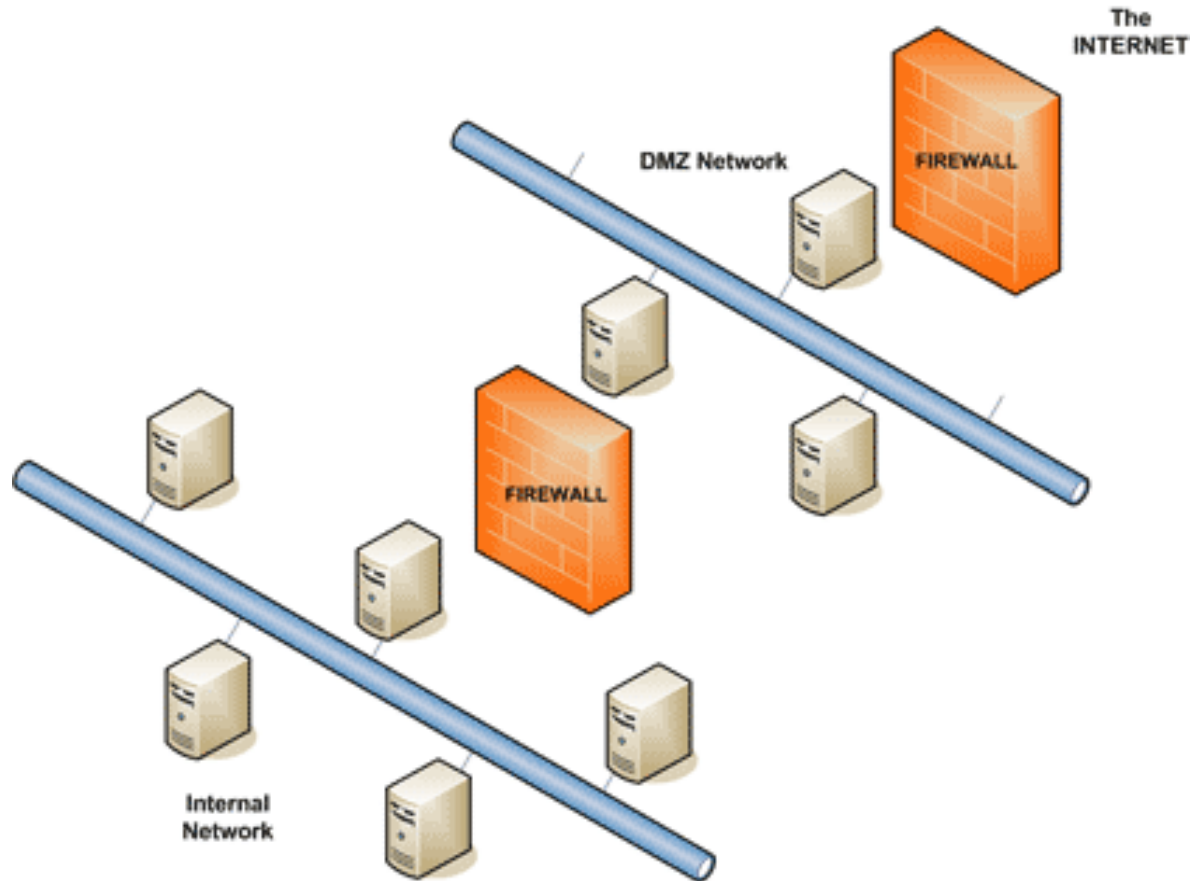
- **Whitelisting**

- Specifies connectivity that is explicitly allowed
 - E.g., allow connections from goodguys.com

Stateless vs. Stateful

- **Stateless:** each packet considered in isolation
- Single packet contains insufficient data to make access control decision
- **Stateful:** allows historical context consideration
 - Firewall collects data over time
 - e.g., TCP packet is part of established session
- *Q: What are the advantages/disadvantages of stateless and stateful?*
 - **Stateful is a lot more effort!**

DMZ (De-militarized Zone)



- Zone between LAN and Internet

Practical Considerations and Limitations

- Network layer firewalls (those operating at the network layer) are dominant
- DMZs allow multi-tiered firewalling
- Tools are widely available and mature
- Personal firewalls gaining popularity
- **Firewall Limitations:**
 - Network perimeters not quite as clear as before: e.g., telecommuters, VPNs, wireless
 - Every access point must be protected
 - Hard to debug, maintain consistency and correctness
 - Often seen by non-security personnel as impediment
 - E.g., Just open port X so I can use my wonder widget

Proxy Firewall

- Firewall variants:
 - *Transparent* : Inline. Allows traffic through. Does not terminate any connection.
 - Proxy: Terminates the connection and raises it up to the application layer, and starts a new connection to the inside.
 - E.g., SSH Gateways into companies.

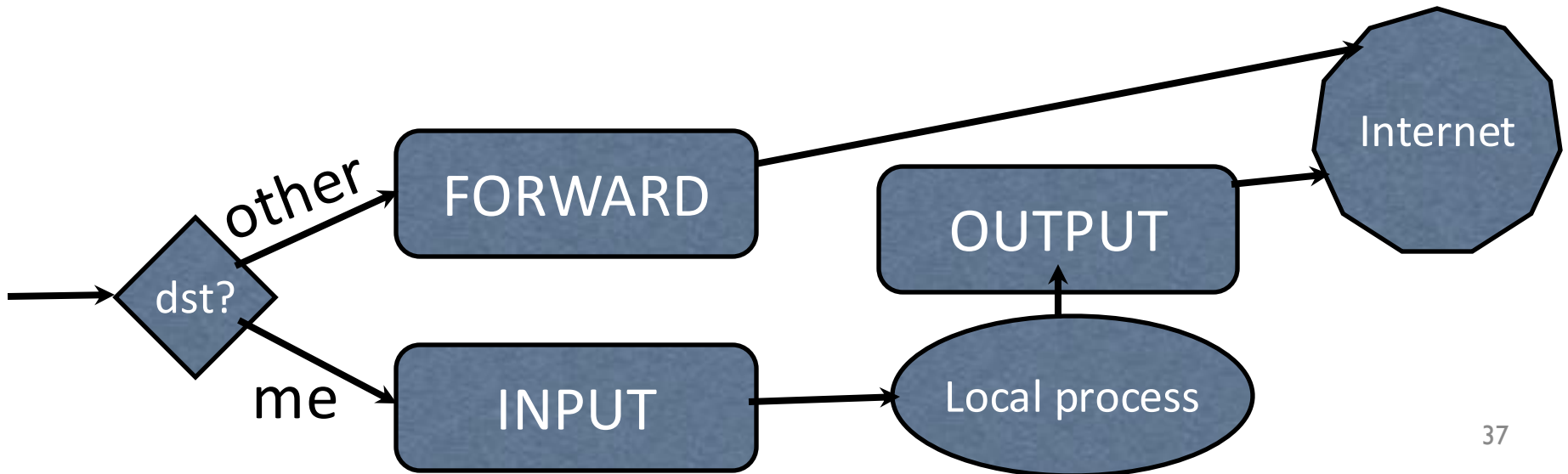
Firewall

Implementations

- Linux **iptables**
[\(http://www.netfilter.org/documentation/HOWTO/packet-filtering-HOWTO.html\)](http://www.netfilter.org/documentation/HOWTO/packet-filtering-HOWTO.html)
- Part of Netfilter
- lots of GUIs, higher-level apps, etc.
- **PF**: OpenBSD Packet Filter
- Mac OS X Application Firewall
- Windows firewall

Netfilter

- Chain: checklist of firewall rules
- Firewall chains:
 - **INPUT**: if packet is destined for this host
 - **FORWARD**: if packet is destined for another network interface
 - **OUTPUT**: outgoing packets
- Packets either get dropped or advance towards next chain



iptables Concepts

The iptables firewall looks in the firewall *table* to see if the *chain* associated with the current *hook* *matches* a packet, and executes the *target* if it does.

- *Table*: all the firewall rules
- *Hook*: each packet has an associated hook
- *Chain*: list of rules associated with the chain id (i.e., the hook name)
- *Match*: when all of a rule's field match the packet
- *Target*: operation to execute on a packet given a match

iptables Rule Parameters

- Non-comprehensive list of things you can match on:
 - Destination/Source
 - Specific IPs, or
 - IP address range and netmask
 - Protocol of packet: ICMP, TCP, etc
 - Fragmented only
 - Incoming/outgoing interface

Per Protocol Options

- Specialized matching options for rules that are specific to a particular protocol
- E.g., for TCP:
 - Source/destination ports (also for UDP)
 - TCP flags: SYN, SYN/ACK, ACK

Targets

- Define what to do with the packet at this time
- **ACCEPT/DROP** (why drop over reject?)
- QUEUE for user-space application
- LOG any packet that matches
- REJECT drops and returns error packet
- RETURN enables packet to return to previous chain

Examples

```
iptables -A INPUT -s 200.200.200.2 -j ACCEPT
```

```
iptables -A INPUT -s 200.200.200.1 -j DROP
```

```
iptables -A INPUT -s 200.200.200.1 -p tcp -j DROP
```

```
iptables -A INPUT -s 200.200.200.1 -p tcp --dport telnet -j DROP
```

```
iptables -A INPUT -p tcp --destination-port telnet -i eth0 -j DROP
```

Deep Packet Inspection

- **Deep packet inspection** looks into the internals of a pack to look for some application/content context
 - e.g., inspect HTTP for URLs that point to malicious websites
 - Can have serious privacy issues if done by, say, Comcast
- To specify a match in iptables
 - `iptables -A INPUT -p tcp -m string --algo bm --string 'exe'`
 - matches packet with content containing 'exe'
 - `iptables -A INPUT -p tcp -m length --length 10:100`
 - matches packet with length between 10 and 100 bytes

Boyer-Moore
string match
algo



Network Intrusion Detection Systems (NIDS)