Report for P538 Project - portScanner

By **Jiecao Chen**

Developers

- Jiecao Chen (jiecchen@indiana.edu)
- Tony Liu (xl41@indiana.edu)

List of Source Files

- **CFileOperation.[h|cpp]** support file I/O operation
- **parse.[h|cpp]** parse the command line arguments
- **Records.h** define a class `Records`, used as processing queue
- **Scanner.[h|cpp]** major class to support ports scan
- **types.h** pre-define several new types
- **utils.[h|cpp]** a various of supporting functions, parse the hostname, output formatting, standard service verifying, etc.
- **portScanner.cpp** main file, multi-threads supports added here

High Level Design

Since we are required to use raw socket, which will recieve packets from all ports, one key problem we need to solve is,

> How can we decide which packet the recieved packet responses to?

The key idea is, for each scan technique (SYN, ACK, NULL, Xmas, FIN, UDP), we use different port to send packets, then we can make decision from the destination port number of the feedback packet, to decide which scan type it responses to.

We keep all `(ip, port, scanType)` triples that need to be processed into a queue, for single thread, we fetch one triples each time, and send a packet to `(ip, port)` with type `scanType`, then wait for the feedback, retransmission is required for lacking of response. We make conclusion for `(ip, port, scanType)` based on the feedback we recieved, and keep that result.

To support **multiple threads**, we set up a map variable `resultsMap` to keep results, which maps the triple `(ip, port, scanType)` to a conclusion we made for this triple. Suppose now we run multiple threads, each is processing a triple, the feedback packet in that thread may not response to the packet we send in that thread, but it does not matter, because when recieve a feedback packet, we simply extract ip (from IP header) port (from UDP, ICMP or TCP header) and scanType (from the port info), make a conclusion and keep the result in to `resultMap`. Also, the object `myRecords` needed to be controled by a mutex variable, otherwise it might be accessed by multiple threads in the same time and cause unpredictable issues.

Key Details

The Scanner Class

Here I put the interface for class Scanner. First we have the definition

```
enum ScanType {SYNScan, NULLScan, FINScan, ACKScan, XmasScan, UDPScan};
enum ResultType {NoResult, NoResponse, Open, Closed, Filtered, Unfiltered, Open_or_Filtered};
```

then the header for Scanner, I also added comments

```
class Scanner {
 public:
```

```
  Scanner();
  void sendPackets(std::string ip, int port, ScanType mode = SYNScan);  // send packets
  bool recvPackets(); // recv and analyze packets
  // test if given ip-port-mode has be analyzed
  bool markedQ(std::string ip_addr, int port, ScanType mode);
  void showResults();  // print out the formated results
  ~Scanner();  // close sockets, release memeory
 private:
  void process_packet(unsigned char* buf, int size);
  void setupRecvSocket(int timeout = 4); // setup a new recv socket, set default timeout = 4
  ResultType analyzeTCPHeader(struct tcphdr *tcph, TPortsDict &portsDict);
  ResultType analyzeICMPHeader(struct icmphdr *icmph, TPortsDict &portsDict);
  ResultType analyzeUDPHeader(struct udphdr *udph, TPortsDict &portsDict);


  struct in_addr source_ip;
  char packet[PACKET_SIZE];  // char[] to represent the packet
  int s; // raw send socket
  int r; // raw recv socket
  struct iphdr *iph; // IP Header
  struct tcphdr *tcph; // TCP Header
  struct udphdr *udph;// udp Header
  TIPDict resultsMap;  // to keep the analyzed results
  TPort2Scan port2scan; // aux array, translate port to scanType
```

};

Major Functionalities in Utils.[h|cpp]

I added comments for each function.

```
// used to calculate the checksum for TCP header
struct pseudo_header /
{
  unsigned int source_address;
  unsigned int dest_address;
  unsigned char placeholder;
  unsigned char protocol;
  unsigned short tcp_length;
  struct tcphdr tcp;
};

// used to calculate the checksum for UDP header
struct pseudo_header_udp
{
  unsigned int source_address;
  unsigned int dest_address;
  unsigned char placeholder;
  unsigned char protocol;
  unsigned short udp_length;
};



// convert hostname/ip to in_addr_t
in_addr_t resolve_to_ip(char *addr);
// convert hostname to ip (char *)
char* hostname_to_ip(char *hostname);


// return the local ip
int get_local_ip(char *);


// calc the checksums
unsigned short csum(unsigned short *ptr,int nbytes);


// used for debug and verbose output
bool printMSG(std::string msg);
// take the same args as printf
// print to standard output when VERBOSE is true
bool printMSG(const char *fmt, ...);


// calc checksum for TCP,
// put in a independent function because might be used by several constructXXXPacket functions
// UDP checksum is calculated directly
void calcTCPCheckSums(struct tcphdr *tcph, struct in_addr source_ip, struct in_addr dest_ip);


// construct packets for different scan types
void constructSYNPacket(struct iphdr*, struct tcphdr*, struct in_addr, struct in_addr, int, int);
void constructNULLPacket(struct iphdr*, struct tcphdr*, struct in_addr, struct in_addr, int, int);
```

```
void constructFINPacket(struct iphdr*, struct tcphdr*, struct in_addr, struct in_addr, int, int);
void constructXmasPacket(struct iphdr*, struct tcphdr*, struct in_addr, struct in_addr, int, int);
void constructACKPacket(struct iphdr*, struct tcphdr*, struct in_addr, struct in_addr, int, int);
void constructUDPPacket(struct iphdr*, struct udphdr*, struct in_addr, struct in_addr, int, int);


// following two are used for verifying standard services
void verifyStdServices(std::string ip_addr);
std::string getServiceName(int port);
```

Known Issues

- DNS for Port 53 has not yet be implemented, I have some problem when tried to fill the DNS query packet.
- It is hard to find an ip-port to test the UDP scan, most ip-port pairs I have tried do not respond to UDP datagram.
- Same situation for ICMP packet, I have not yet recieved any ICMP feedback so far. Have check the code for many times, not sure if there is a bug.

Reference

- http://en.cppreference.com/
- http://www.binarytides.com/tcp-syn-portscan-in-c-with-linux-sockets/
- http://www.binarytides.com/raw-udp-sockets-c-linux/
- http://en.wikipedia.org/ for TCP UDP ICMP headers
- Computer Networking: A Top-Down Approach (6th Edition)
- http://stackoverflow.com/