Comp 8505 Assignment 4

A01053901 Geoffrey Browning

Contents

Purpose

Learn to use and apply covert channels for transferring data with raw sockets.

Requirements

Requirement	Status
Sender utilizes raw sockets to send data hidden in covert channels.	FULLY IMPLEMENTED
Receiver extracts and displays data from fields	FULLY IMPLEMENTED

Platforms

Works on Linux and Windows operating systems.

Language

Python3

Design

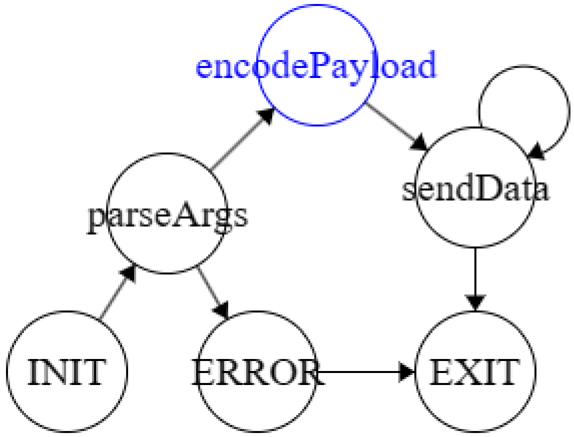
Sender

FSM

State Table

From State	To State	Action
INIT	argParse	Parse Args
argParse	Error	Argument Error
argParse	encodePayload	Payload Encoding
encodePayload	sendData	Send Data over ICMP
sendData	sendData	Send Data in loop till done
sendData	Exit	Exit Program
Error	Exit Exit Program	

State Transition Diagram



Pseudocode

```
Def argParse(argv)
    Initialize ip to None
    Initialize msg to None
    parse args for ip and msg
    return ip, msg


Def encodePayload(payload)
    stepOne = ‘ ‘.join(format(ord(c), ‘08b’) for c in payload)
    stepTwo = [stepOne[c:c+8] for c in range(0, length(stepOne), step by 8)]
    return stepTwo


Def sendData(data)
    payload = encodePayload(data[1])
    Initialize ipLayer to IP header with destination data[0]
    for p in payload:
        Initialize icmpLayer to ICMP header with type=int(p, base=2))
        Initialize icmpPacket to ipLayer / icmpLayer
        send(icmpPacket, verbose=true)


Def main(argv)
    ip, msg = argParse(argv)
    sendData([ip, msg])
```

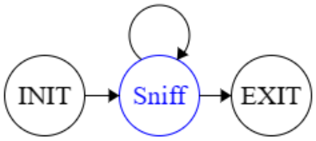
Receiver

FSM

State Table

From State	To State	Action
INIT	Sniff	Sniff for ICMP packets
Sniff	Exit	Exit program

State Transition Diagram



Pseudocode

```
Def packetHandler(packet)
        if packet is ICMP packet
                src = packet[IP].src
                dst = packet[IP].dst
                type = packet[ICMP].type
                print(Source: src Destination: dst Type: int(type) Decoded Payload: chr(type)
Def main()
        sniff(prn=packetHandler)
```

Testing

Test	Result	Expected
Send 1234567890 to receiver	1234567890 is displayed one character after another	1234567890 is displayed one character after another
Send helloworld to receiver	helloworld is displayed one character after another	helloworld is displayed one character after another
Send HelloWorld to receiver	HelloWorld is displayed one character after another	HelloWorld is displayed one character after another
Send ^^^^^^ to receiver	^^^^^ Characters are displayed one after another	^^^^^ Characters are displayed on after another
Run Sender without command line arguments	Gracefully exits after displaying error message for no command line arguments	Gracefully exits after displaying error message for no command line arguments

User Guide

```
Sudo python geoReceiver.py
Sudo python geoSender.py -h <target ip> -m <Msg to send>
```

Discussion

Tables detailing the fields:

Fields:

Field	Reasoning
Version	Not suitable for a covert channel as it is a fixed field indicating version.
Header Length	Not suitable as it is a fixed field indicating length of ip header.
Type of Service	Potentially suitable but not commonly used.
Total Length	Not suitable, cannot be easily manipulated without disrupting a packet.
Identification	Suitable, commonly used. Can be easily modified to carry data.
IP Flags	Potentially suitable but not commonly used.
Fragment Offset	Manipulation could disrupt packet fragmentation.
Time to Live	Suitable, often used for covert channels.
Protocol	Suitable, used in some covert channels.
Header Checksum	Modifying this would likely lead to packet rejection.
Source Address	Not suitable as it is a fixed field and easily detected.
Destination Address	Not suitable as it is a fixed field
IP Options	Potentially suitable if data is hidden in less frequently inspected fields.
Traffic Class	Potentially suitable but rarely used.
Flow Label	Fixed field used for flow identification.
Payload Length	Altering this could disrupt packet integrity.
Next Header	Can be altered to convey information about the type of covert data.
Hop Limit	Potentially suitable, often used.
Source Port	Suitable for covert channels in UDP and TCP.
Destination Port	Suitable for UDP and TCP covert channels.
Sequence Number	Potentially suitable but the sequence number must always go up.
Acknowledgement Number	Potentially suitable but can disrupt proper TCP operations.
Data Offset	Not commonly used, could disrupt packet processing.
Reserved	Not suitable.
Flags	Specifically reserved, urgent, ack, push, reset, and syn can be manipulated.
Window	Potentially suitable but not commonly used.
Checksum	Altering this would likely lead to packet rejection.
Urgent pointer	Not frequently used but could be suitable.
Options+Padding	Potentially suitable in conveying information.
Data	Highly suitable and commonly used to hide information within data.
UDP Checksum	Modifying this would likely lead to packet rejection.
Type	Can be altered to carry hidden data.
Code	Can be altered to carry hidden data.
ICMP Checksum	Modifying this would likely lead to packet rejection.

Usable Fields Ranked Best to Worst:

Field
Data (TCP/UDP)
Source Port (TCP/UDP)
Destination Port (TCP/UDP)
Flags (TCP)
Protocol (IPv4)/Next Header(IPv6)
Identification (IPv4)
Time to Live(IPv4)/Hop Limit (IPv6)
Type of Service (IPv4)/Traffic Class(IPv6)
Options+Padding (TCP)
Type (ICMP)
Code (ICMP)
IP Flags (IPv4)
IP Options
Sequence Number
Acknowledgement Number
Urgent Pointer