Comp 8505

Selected Topics in Network Security Development

Assignment Two

Dean Morin – A00750619

Introduction

The purpose of this program is to run a backdoor application that sniffs incoming traffic for a particular signature. Packets matching the signature will contain an encrypted command to run.

Design

The backdoor will sniff all UDP traffic on port 34249. If the source port on one of those packets equals the port returned from port_from_date(), then the contents of the packet are decrypted, with the results fed into a system() call.

The backdoor is required to run with root privileges so that it can sniff network traffic using libpcap.

Process Name

In order to hide that the program is running from nosy administrators, the first thing that we'll want to do is disguise the process name. I went with a standard Linux daemon, just in a different directory than where it's usually found. I used

    /usr/libexec/udevd

This daemon is normally found at

    /sbin/udevd

The aim is to use a name that won't look out of place or maybe even somewhat familiar with, but not familiar enough to notice that anything is out of the ordinary.

Snap Length

When setting up a pcap session, there is an option to limit the amount of data captured for each packet. Experience with tcpdump has shown that the kernel can drop packets in times of very heavy traffic or on slower systems (resulting in those packets not being sniffed). Limiting the snap length can reduce the number of dropped packets, so the snap length value will be set to the minimum amount needed to read in the packet headers and the maximum command message size.

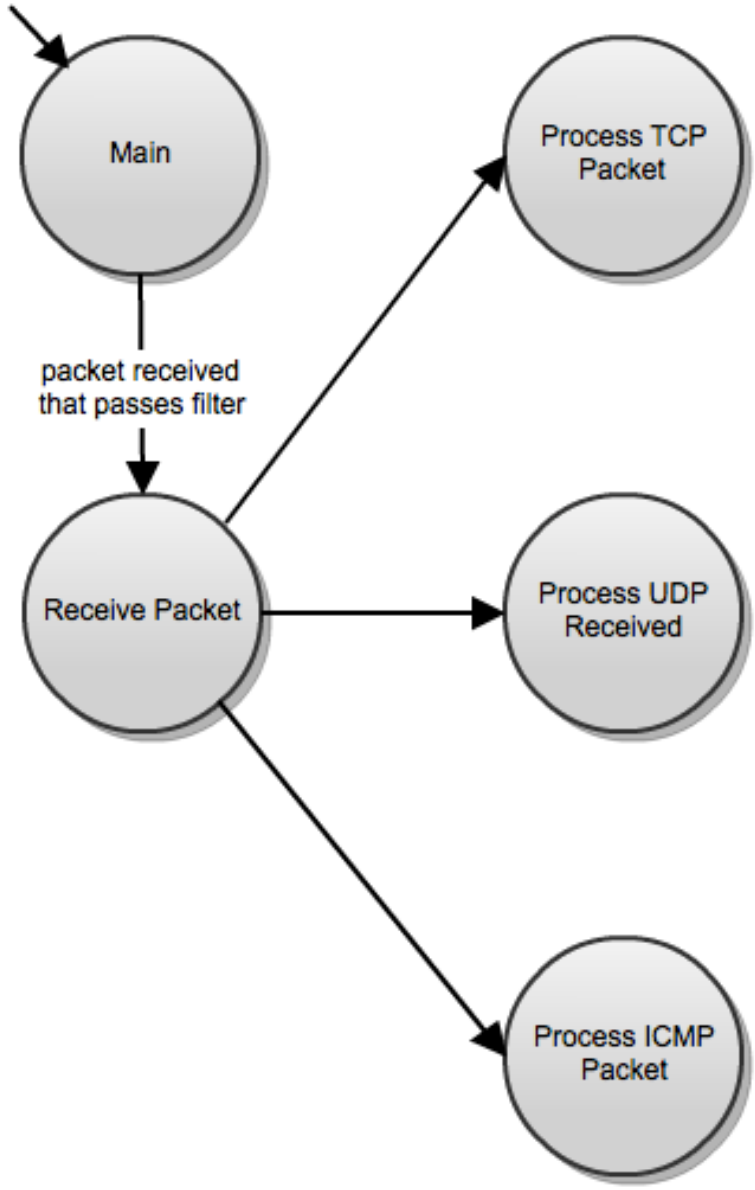
Encryption

The encryption method I'll use is a block cipher called "XTEA" (eXtended Tiny Encryption Algorithm). It's a very lightweight solution that provides quite a

reasonable amount of security. It uses symmetric keys, which is important since I want to avoid key exchanges.

The key will be stored in the backdoor binary. To make discovering it in the binary more difficult, the key will be stored in "mangled" form (extra characters and in the wrong order). Each time the key is needed, the stored key constant will be run through an algorithm to extract the actual key. Immediately after use, the array storing the key will have memset() called on it so that the key isn't sitting in memory.

State Transition Diagram



Psuedocode

Main

- change process name
- set up pcap session with filter “udp dst port 34249”
- pcap loop

Receive Packet

- check the transport protocol and call the related handler function (TCP and ICMP will be stubbed out for this assignment)

Process TCP / ICMP Packet

- stubbed out

Process UDP Packet

- return if the source port does not match the port returned from port_from_date()
- get the length of the message from the IP ID field (do not change endianess from network order)
- unencrypt message to reveal command
- append “ &> /dev/null” to the command to make sure the results aren’t printed to screen when the command is run
- run unencrypted command with system()

Instructions & Testing

Building

There are two binaries that need to be built:

1. udevd– the sniffing application
2. client_util – used by the testing script "test.sh" in conjunction with hping3 to send a test packet to the backdoor

The backdoor should be built with "make debug" for demonstration purposes, as the regular version shows no output. After that, "chown root" and "chmod +s" need to be run.

```
~/Dropbox/c8505/assign2$ make debug && sudo chown root ./backdoor && sudo chmod +s ./backdoor
gcc -W -Wall -pedantic -g -DDEBUG -c backdoor.c
gcc -W -Wall -pedantic -g -DDEBUG -c pkthdr.c
gcc -W -Wall -pedantic -g -DDEBUG -c xtea.c
gcc -W -Wall -pedantic -g -DDEBUG -c util.c
gcc -W -Wall -pedantic -g -DDEBUG -lpcap -o backdoor backdoor.o pkthdr.o xtea.o util.o
```

(Please note that these screenshots were taken when the backdoor binary was called "backdoor" rather than "udevd")

The client_util application can be built with "make client_util."

Running

First, run the backdoor application. Once it's running, run the test script as su. It takes two arguments, the destination IP and the command to be run.

```
~/Dropbox/c8505/assign2$ sudo ./test.sh localhost "ls -la"
HPING localhost (lo 127.0.0.1): udp mode set, 28 headers + 100 data bytes
[main] memlockall(): Success
Warning: can't disable memory paging!
ICMP Port Unreachable from ip=127.0.0.1 name=localhost.localdomain

--- localhost hping statistic ---
1 packets tramitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
```

After sending a packet, you can see the command executed by the backdoor progam.

```
~/Dropbox/c8505/assign2$ ./backdoor


        0x0000:  0000 0304 0006 0000 0000 0000 0000 0800  ................
        0x0010:  4500 0080 0600 0000 4011 766b 7f00 0001  E.......@.vk....
        0x0020:  7f00 0001 d3e8 85c9 006c 6eef eede 179c  .........ln.....
        0x0030:  9671 9185 0a00 0000 0000 0000 0000 0000  .q..............
        0x0040:  0000 0000 0000 0000 0000 0000 0000 0000  ................
        0x0050:  0000 0000 0000 0000 0000 0000 0000 0000  ................
        0x0060:  0000 0000 0000 0000 0000 0000 0000 0000  ................
        0x0070:  0000 0000 0000 0000 0000 0000 0000 0000  ................
        0x0080:  0000 0000 0000 0000 0000 0000 0000 0000

        Command: "ls -la"

total 368
drwxrwxr-x.  3 dean dean  4096 May 29 01:23 .
drwxrwxr-x. 10 dean dean  4096 May 22 18:30 ..
-rw-rw-r--.  1 dean dean 30907 May  8 18:47 Ass2-12.pdf
-rwsrwsr-x.  1 root dean 24301 May 29 01:23 backdoor
-rw-rw-r--.  1 dean dean  5321 May 29 00:50 backdoor.c
```

Finally, use ps or htop to confirm that the process name has been correctly altered.

```
  1  [|||                                                                    3.2%]     Tasks: 93,
  2  [|||                                                                    3.3%]     Load avera
  3  [                                                                       0.0%]     Uptime: 5
  4  [                                                                       0.0%]
  Mem[|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||772/3954MB]
  Swp[|                                                                   20/6015MB]

  PID USER      PRI  NI  VIRT   RES   SHR S CPU% MEM%   TIME+  Command
 3609 root       20   0  165M  5360  4388 S  0.0  0.1  0:00.00 /usr/libexec/packagekitd
 3607 root       20   0  165M  5360  4388 S  0.0  0.1  0:00.02 /usr/libexec/packagekitd
  846 root       20   0  208M  5276  3644 S  0.0  0.1  0:01.81 /usr/libexec/polkit-1/polkitd --no-debug
  841 root       20   0  208M  5276  3644 S  0.0  0.1  0:02.67 /usr/libexec/polkit-1/polkitd --no-debug
 1494 dean       20   0   97M  2300  1832 S  0.0  0.1  0:00.00 /usr/libexec/pulse/gconf-helper
 1288 rtkit      RT   1  160M  1224  1044 S  0.0  0.0  0:01.40 /usr/libexec/rtkit-daemon
 1287 rtkit      20   0  160M  1224  1044 S  0.0  0.0  0:02.93 /usr/libexec/rtkit-daemon
 1280 rtkit      21   1  160M  1224  1044 S  0.0  0.0  0:04.36 /usr/libexec/rtkit-daemon
 1543 dean       20   0  335M  6252  4536 S  0.0  0.2  0:00.00 /usr/libexec/tracker-miner-flickr
 1544 dean       20   0  335M  6252  4536 S  0.0  0.2  0:00.32 /usr/libexec/tracker-miner-flickr
 1539 dean       20   0  335M  6252  4536 S  0.0  0.2  0:00.42 /usr/libexec/tracker-miner-flickr
 1615 dean       20   0  443M  7624  5388 S  0.0  0.2  0:00.00 /usr/libexec/tracker-miner-fs
 1616 dean       20   0  443M  7624  5388 S  0.0  0.2  0:00.45 /usr/libexec/tracker-miner-fs
 1600 dean       39  19  443M  7624  5388 S  0.0  0.2  0:04.07 /usr/libexec/tracker-miner-fs
 1549 dean       20   0  547M 11284  4788 S  0.0  0.3  0:00.00 /usr/libexec/tracker-store
 1621 dean       20   0  547M 11284  4788 S  0.0  0.3  0:00.00 /usr/libexec/tracker-store
 1622 dean       20   0  547M 11284  4788 S  0.0  0.3  0:00.00 /usr/libexec/tracker-store
 1623 dean       20   0  547M 11284  4788 S  0.0  0.3  0:00.15 /usr/libexec/tracker-store
 1551 dean       20   0  547M 11284  4788 S  0.0  0.3  0:01.04 /usr/libexec/tracker-store
 1620 dean       20   0  547M 11284  4788 S  0.0  0.3  0:03.74 /usr/libexec/tracker-store
 1546 dean       20   0  547M 11284  4788 S  0.0  0.3  0:05.74 /usr/libexec/tracker-store
 3464 root       20   0  8996  3072  2968 S  0.0  0.1  0:00.00 /usr/libexec/udevd
 1515 root       20   0  118M  3116  2452 S  0.0  0.1  0:00.00 /usr/libexec/udisks-daemon --no-debug
 1513 root       20   0  118M  3116  2452 S  0.0  0.1  0:00.46 /usr/libexec/udisks-daemon --no-debug
 1234 root       20   0  150M  4028  2876 S  0.0  0.1  0:00.00 /usr/libexec/upowerd
 1235 root       20   0  150M  4028  2876 S  0.0  0.1  0:00.01 /usr/libexec/upowerd
 1233 root       20   0  150M  4028  2876 S  0.0  0.1  0:31.62 /usr/libexec/upowerd
  894 root       20   0  174M  6420  4888 S  0.0  0.2  0:00.01 /usr/sbin/NetworkManager --no-daemon
  839 root       20   0  174M  6420  4888 S  0.0  0.2  0:00.07 /usr/sbin/NetworkManager --no-daemon
  769 root       20   0  174M  6420  4888 S  0.0  0.2  0:03.89 /usr/sbin/NetworkManager --no-daemon
```

Additional Testing

To make sure that the backdoor is correctly discriminating based on signature, I sent messages with different source or destination ports. Also, many commands of varying lengths were tried to ensure that the encryption and decryption was working as expected. Finally, commands that exceeded the determined limits were correctly rejected by the client application.

Areas to Improve

There are some weaknesses with the current implementation of the backdoor:

1. It can be defeated by a simple restart of the machine.
2. It has an associated TTY in the process list, even when run in background mode.
3. An astute system administrator could see that the process name does not belong.

The obvious answer to these issues would be to make the backdoor application a daemon. For a really effective backdoor, it would be best to modify the source of an existing daemon to include the desired functionality.

One thing that would have to be carefully considered is how to avoid issues with the system's package manager.