Backdoor – Design and Test Data

Andrew Maledy

COMP8505 – Assignment 2

Prepared for Aman Abdulla, April 30th, 2013.

Table of Conents

Pseudocode 3

*Client* 3

*Server* 3

*Application Design (demonstrated by intended use-case)* 4

Pitfalls & Rationale 5

Testing (Packet Capture) 5

Testing (Client/Server file stats) 9

# Pseudocode

## Client

Use Case: “Send a command to a compromised host (which then executes the command and returns the output to the sender).”

* Get options {source\_ip, dest\_ip, source\_port, dest\_port, interface}.
* Mask process (something client-like).
* Prompt user for command
  + Prepare headers
  + Foreach byte on command, put byte in TCP Window and send packet. When command is done, send FIN packet.
  + Listen for remote host to respond – and display output
    - (Future release: timeout functionality?)
  + Return to prompt()

## Server

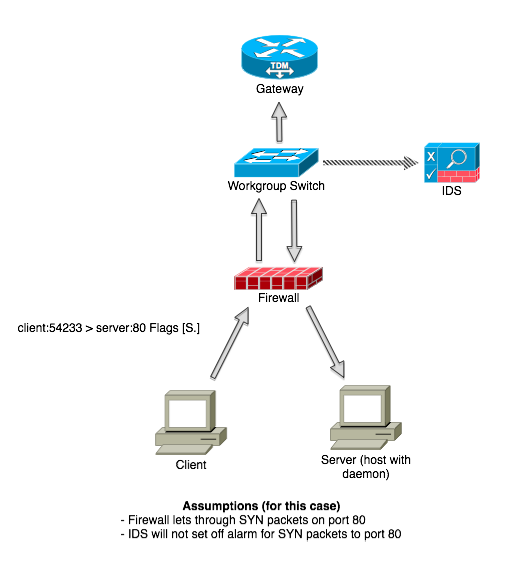
Use Case: “Run a daemon-like tool to listen commands from a remote host”

* Get options {source\_ip, source port}.
* Mask process (something server-like)
* Uusing Libpcap, listen for packets from the remote host.
  + When packets come in on the specified port, concatenate the TCP Window field until a FIN comes down.
  + Execute the command and grab the output
  + Send the output back to the client, followed by a FIN.
  + Return to listen()

# 

# *Application Design (demonstrated by intended use-case)*

This application works on the same principles as the covert channels assignment in that it hides data in unusual headers as to bypass firewalls and slide under IDS. A downside to this method of transport is speed. A simple command such as `ls` can take a second or so to get data back. In addition, this method renders TCP connectionless, which runs the risk of packets not arriving or even arriving out of order.



Process masking us a great way to slip under the radar of an unsuspecting target.

# Pitfalls & Rationale

* No reliability built in.
  + Reliability is dangerous. We don’t want people to know that there is data coming through.
* Speed (byte by byte can take time).
  + We must send small amounts of data at a time to avoid detection.
* Not one stops solution and may not be suitable for all instances. Ie. SYN requests to a web server are not out of place. Syn requests to a log server or administrative console could be.
* Requires a compromised host (with the server running).

# Testing

The methods used for covert channels are exactly the same as those used in assignment 1 (covert file transfer). Hiding data in the TCP Window field and setting the SYN flag allows packets to seamlessly flow through firewalls and bypass IDS perimeters that may be in place. Confident that our methods our covert, let us examine process masking to prove our tool is running in a disguised manner.

**Process Masking**

The code to mask a process in ruby is very simple.

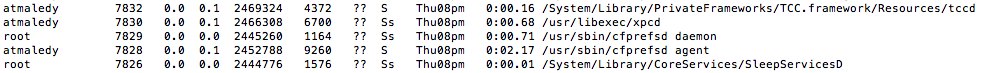
$0 = “my\_backdoor”

Obviously “my\_backdoor” is not a process mask one would ever use as that would be a dead giveaway however, for proof of functionality we’ll set it to this and take a look at our process table. Let’s examine the output:

root 8704 0.0 0.0 2432784 668 s000 R+ 10:35am 0:00.00 my\_backdoor

If anybody saw the above code they would hopefuly stop the process and wipe their machine. Let’s examine how this might look with a more subtle process mask.

$0 = “xpcd”



The second row on that list is our backdoor. The only way to catch this is to notice that there are two instances of xpcd running and with hundreds of processes, it is not feasible for a user to be constantly checking for duplicates. We are now running in a covert-manner on the comomised machine.

# Testing (Client/Server file stats)

**Client side file information collected from client and server**

Vm1-fedora1:Assignment1 atmaledy$ stat secret\_message.txt

16777218 3003852 -rw-r--r-- 1 atmaledy staff 0 20 "Apr 29 16:28:24 2013" " Apr 29 16:28:24 2013" "Apr 29 16:28:24 2013" "Apr 29 17:24:20 2013" 4096 8 0x40 secret\_message.txt

**Server side**

Vm2-fedora:Assignment1 atmaledy$ stat secret\_message.txt

16777218 3003852 -rw-r--r-- 1 atmaledy staff 0 20 " Apr 29 16:30:6 2013" "Apr 29 16:30:06 2013" "Apr 29 16:30:07 2013" "Apr 29 16:30:07 2013" 4096 8 0x40 secret\_message.txt

**Here you can see the date created/modified matches up with the packet capture. The file size from the client and server also match accordingly.**