Backdoor – Design and Test Data

Andrew Maledy
COMP8505 – Assignment 2
Prepared for Aman Abdulla, April 30th, 2013.

Table of Conents

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()

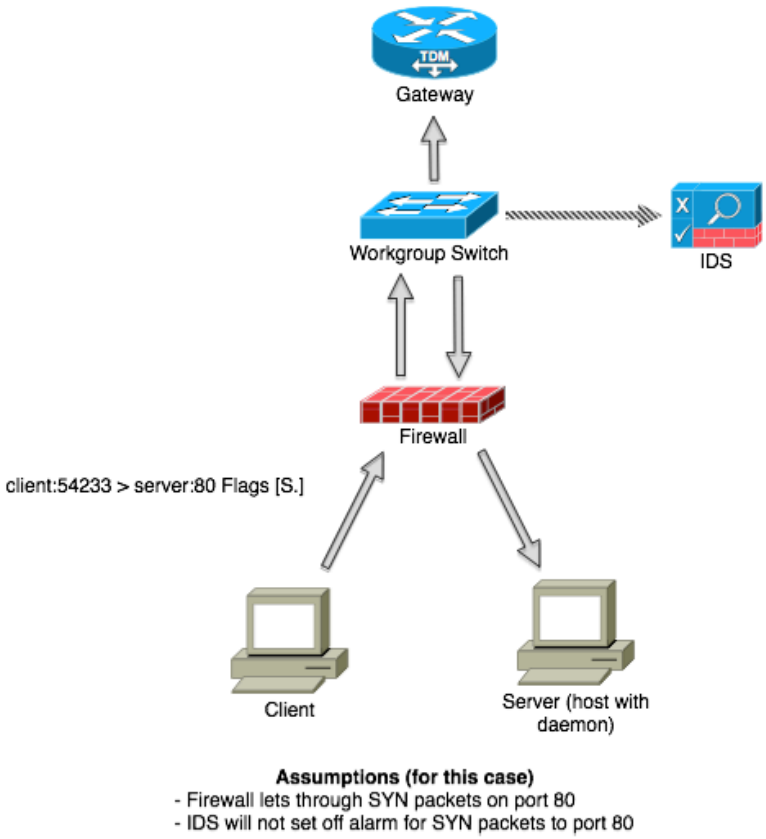
Server

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.

Mask backdoor as a non suspicious process (Pass)

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"
```

```
atmaledy       7832   0.0  0.1  2469324   4372   ??  S    Thu08pm   0:00.16 /System/Library/PrivateFrameworks/TCC.framework/Resources/tccd
atmaledy       7830   0.0  0.1  2466308   6700   ??  Ss   Thu08pm   0:00.68 /usr/libexec/xpcd
root           7829   0.0  0.0  2445260   1164   ??  Ss   Thu08pm   0:00.71 /usr/sbin/cfprefsd daemon
atmaledy       7828   0.0  0.1  2452788   9260   ??  S    Thu08pm   0:02.17 /usr/sbin/cfprefsd agent
root           7826   0.0  0.0  2444776   1576   ??  Ss   Thu08pm   0:00.01 /System/Library/CoreServices/SleepServicesD
```

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.

Test 2 - List files and directories of compromised host (Pass)

This test is to determine whether or not the server is receiving and executing a command as well as returning the output to the client. Data is transferred over a covert channel.

Client side file information collected from client and server

Server (folder contents italicized)

```
Vm2-fedora1:Assignment2 ls

     Top_secret_file.txt Passwords Missile_launch_codes

Vm2-fedora1:Assignment2 atmaledy$ ruby src/server.rb
```

Client (server output italicized)

```
Vm1-fedora1:Assignment2 atmaledy$ ruby src/client.rb
Enter a command > ls
Remote Server Says:

Top_secret_file.txt
Passwords
Missile_launch_codes
```

Above you can clearly see that the server has sent the data returned by the client-issued-command back. We can see that the client has displayed it accordingly.