Lab 15.1: Some Basic Forensic Analysis
IN618 Security

June 11, 2015

Introduction

If we suspect a system has been compromised we need to do a little forensic analysis to se if this has, in fact, happened. This is complicated somehwat because a skillful attacker will modify system utilities to conceal his or her activity on the compromised host. On the other hand, if we can determine that these utilities are modified, that is strong evidence that a system is compromised.

1 "Compromise" your system

Log into your Linux server as instructed by the lecturer. On that system, download and prepare our "malicious code" by carrying out the following steps:

```
curl http://kate.ict.op.ac.nz/~tclark/malicious-server > malicious-server
chmod +x malicious-server
cat /etc/passwd > data
```

Our `malicious-server` script is just a simple one-line shell script that echoes the contents of the file `data` out on a network socket. We're having it show the contents of `/etc/passwd`. Run your script in the background with the command.

```
./malicious-server &
```

When you do this, the terminal will output the *process id*, or `PID` of your running process. Note that number here. You will need it later.

Verify that your program is running with the command

```
telnet localhost 8085
```

You should be served the contents of the `passwd` file you captured earlier.

Now, we'll cover our tracks a bit by removing our malicious script with the command

```
rm malicious-server
```

2 Examining our system

2.1 Find the flie

We deleted our malicious script. Because it's still running, we can see where the file used to be. The command `lsof` shows us all open files (and file-like objects) on the system. Run the command. It shows a lot of output,

and in a real situation we would have to go through it to find and investigate anything that looks suspicious. In our case, however, we can short circuit the process with the command.

```
lsof | grep malicious-server
```

This technique, removing a malicious program's file after starting it, is an easy way for an attacker to hide activity but since the process is still in memory we can still find the evidence.

2.2 Look for network connections

Our malicious script listens for network connections, so we can look for that as well with the `netstat` command. If we want to look for active network network connections we can use the command

```
netstat -ta
```

to check for TCP and UDP activity. In our case we see that we have something listening on port 8080, or `http-alt` that we don't expect.

Earlier this semester we also saw the we could use port scanning to look for open network sockets remotely. There are at least two reasons why port scanning is advantageous in this situation. Can you think of what they are?

2.3 Look for the malicious application

There are a few ways we can try to find our malicious server application while it is running.

`ps` is a command that shows running processes. If we run `ps -A` we can see all the running processes and look for anything that is out of place. In our case, our "attacker" chose a pretty suspicious sounding name for the malicious process. Real atackers will probably be a litle more cunning. What are some ways we can identify suspicious processes.

`top` is another tool that shows runnig processes. It ususally runs interactively and updates every few seconds. Try it and see if you can find your malicious process. Often, `top` shows more data than can fit on one screen, so running `top -b -n 1` will give you a snapshot of output that you can inspect more easily.

`ps` and `top` get their information by looking at the `proc` file system. There is a directory under `proc` for every running process, identified by the PID. Find the `/proc` subdirectory for the PID of your malicious process and cd into it. Use `ls -l` to list its contents and then use `cat status` to see information about the running process.

Sometimes on a compromised system you may find a process under the `/proc` that doesn't show up in output from `top` or `ps`. What might that mean?