Hawk Host Blog

All things Hawk Host

Using netcat as an intercepting proxy

Posted on December 12, 2009 by Cody

A couple of days ago I needed a way to see how a particular client was interacting with a server. Obviously there are numerous ways to do this, but I was curious how easy it would be to implement something similar with a quick netcat command. Sure enough after a little bit of fiddling I was able to produce exactly what he needed.

> *nc -l -p 12345 < pipe | tee outgoing.log | nc server 12345 | tee pipe incoming.log*

Now this may seem a little cryptic so I'm going to dissect each portion to explain how it works. Keep in mind the "pipe" references an actual pipe. You can make a FIFO pipe by running "mkfifo pipe" or "mknod pipe p" – the former is the most usual way. If you're not familiar with named pipes I recommend reading up on them before continuing with this post as you may get a little confused.

> *nc -l -p 12345 < pipe*

This portion simply has netcat listen on port 12345 and send anything from the pipe to the connected client. If you're not familiar with the pipes think of it as a simple file with the word "hello" in it. When someone were to successfully connect to the netcat instance it would send the "hello" to the client.

> *| tee outgoing.log*

If you're not familiar with tee this may seem a bit obscure. Tee prints the things piped to it to stdout as well as writing it to a file. In this instance any traffic from the connected client will get printed to stdout and to the file "outgoing.log". An example of how this would work is if I connected to the netcat instance and simply typed "hello" it would print it out to the screen and log it to the "outgoing.log" file.

> *| nc server 12345*

This is the server that you would normally want to connect to. Remember the goal is to make a quick intercepting proxy to see how the client reacts to the server. This is the server.

> *| tee pipe incoming.log*

Here is where the magic happens. This completes the relay so the client and server can communicate across the proxy. What this does is takes the network traffic from the server and using tee prints it to stdout while piping it to our "pipe" and "incoming.log" files respectively.

Now all of this may make sense individually, though how they work together might be slightly confusing.

If you recall the first command sends all data from our “pipe” to the client – and at the end we pipe all data from the server to the “pipe”. See now? We’re simply taking all data the server send and sending it to the client completing the relay and allowing for normal operation.

Now in this case I needed this for a quick look at how a normal IRC client interacts with the server since the RFC is lacking – so here is a real world example of where this was used (though there’s likely infinite better ways to do it):

```
nc -l -p 12345 < pipe | tee outgoing.log | nc irc.freenode.net 6667 | tee pipe incoming.log
```

You’ll notice when you execute the above command you’ll start seeing some traffic from the server instantly:

```
NOTICE AUTH :*** Looking up your hostname…
NOTICE AUTH :*** Checking ident
NOTICE AUTH :*** No identd (auth) response
NOTICE AUTH :*** Couldn’t look up your hostname
```

Now we connect to the netcat server – in this case localhost on port 12345 and if everything goes as planned it should connect like normal to Freenode. If you take a peak at the netcat server you’ll see a bunch of activity!

The cool part is the logs – we can see exactly how this particular IRC client (IRSSI) and server (Freenode) interact.

incoming.log
outgoing.log

Once again this isn’t the best way to do this – tcpdump, wireshark and infinite other choices are available. That being said it’s fun to fiddle and learn.

-Cody

This entry was posted in Tips and tagged linux, Tips, tricks. Bookmark the permalink.

4 Responses to *Using netcat as an intercepting proxy*

Harry *says:*

February 17, 2011 at 12:04 am

Cody, That was a very lucid explanation, thanks! I wish you had covered (or, could now cover!) use of netcat with https. I only discovered yesterday that netcat does not talk https and that you must employ stunnel etc.

Kristian Nygaard Jensen *says:*

November 29, 2012 at 9:42 am

The only way I can make your example work is to add a >(>) before incomming.log.

And in my version of nc the -p option should be omitted in the first call to nc.

But nice how-to

Kristian Nygaard Jensen *says:*

November 29, 2012 at 9:51 am

A comment on my comment the weird looking >(>) is just meant as a > but I didn't know how the editor would handle the > character so i used the & gt ; code. Would you please edit my comment and just write a single >

Kind Regards
Kristian

Perro *says:*

March 12, 2013 at 6:09 pm

Not very useful for HTTP requests as once the connection is closed by either side the pipe will be broken because nc terminates as soon as a connection is closed.

Hawk Host Blog
Proudly powered by WordPress.