Security Analysis with Wireshark

By Russ McRee

It's not likely that you'll run into someone working in information security who hasn't heard of Ethereal.

As of May 2006, the world's most popular network protocol analyzer has become Wireshark. Gerald Combs, Ethereal's creator, was unable to reach agreement with his now former employer, which holds trademark rights to the Ethereal name. However, Combs wisely maintained ownership of the source code, and so Wireshark is, for all intents and purposes, Ethereal continued. Wireshark.org offers all the details of the name change and the Ethereal/Wireshark history, so we'll not cover any more of that here.

Understanding how to use this tool is certainly extremely useful in the infosec workplace, and essential for classes like SANS' Security 503: Intrusion Detection In-Depth, or most anything Laura Chappell's Protocol Analysis Institute offers. Wireshark/Ethereal is considered invaluable to network engineers for troubleshooting and analysis; and for security analysts there is deeply revealing insight at your fingertips. Keep in mind that, as we discuss Wireshark, the vast majority of our content is also immediately relevant to older versions of Ethereal.

As we proceed through our discussion we'll assume, given the widespread use and ease of installation, that you're running Wireshark on Windows.

Security issues

Before diving into Wireshark use scenarios, let's review some security issues innate to the tool itself.

First and foremost, the sheer nature of Wireshark makes it more vulnerable than other software, in part because it's written in ANSI C, and “the developers providing code to Wireshark ... have very divergent programming experience, from advanced networking specialists to novice programmers, making it more likely that new bugs get in”.

From the Wireshark wiki come four extremely valuable points:

- Always update to the latest Wireshark version available as bugs are fixed often.
- Analyze capture files in an uncritical environment. Create a special (limited) user account or even use a dedicated machine for this task.
- Use a small capture tool which is less likely affected by security bugs, e.g.: tcpdump and transfer the capture file to the uncritical environment mentioned above.

There's no better way to persuade them than by showing how easy it is to pull their credentials off the wire.

A simple scenario with Wireshark: Down with cleartext

We've all seen it: legacy servers and apps, older network devices, FTP, email apps. What information security evil do they all share? Cleartext protocols, one and all. Imagine the old-school DBA or legacy systems manager who has a hard time understanding what could possibly be wrong with telnet or R commands. Let's assume you have a SPAN port with a tap or a hub for network analysis, and have management buy-in to sniff traffic. You've been mandated with clearing out cleartext protocols, but need to drive the point home for hardcore legacy users upset by the premise of having to use an SSH client. There's no better way to persuade them than by showing how easy it is to pull their credentials off the wire.

http://wiki.wireshark.org/Security

Footnotes:
1 http://en.wikipedia.org/wiki/Wireshark
2 http://wiki.wireshark.org/Security
3 http://wiki.wireshark.org/Security
Please, above all else, be certain you have approval to conduct this exercise. In the wrong circles, this activity can get you in a lot of trouble. When I’ve done this in the past, I’ve always informed the user in question of my intent and asked that they use a non-critical account or temporarily change their password, in the understanding that you’re going to hand it back to them.

The recommended method is to capture the traffic with a smaller, safer tool like tcpdump or WinDump. Yes, we’re focused on Wireshark, but you need to be proficient with basic captures too, so here’s a crash course in tcpdump/WinDump. Get WinDump here: http://www.winpcap.org/windump/install/.

Assuming a Windows system is attached to the hub on your SPAN port, execute windump -D from a command prompt. This will tell you what interfaces are available. The interfaces will be sequentially numbered if there are more than one. For our example we’ll use interface 2.

Issue the command windump -i 2 > cleartext.pcap.

This will write the capture to cleartext.pcap which you can then review in Wireshark. Have your user log on to their favorite legacy system via telnet while you are running the capture. Once they’ve completed a quick process, or just logged on or off, stop the WinDump capture and move the cleartext.pcap file to your machine where you’re running Wireshark.

If you’d like to conduct the analysis part of this scenario as you read, pretend that a sample capture available on the Wireshark wiki is the file you’ve captured.

Grab the telnet-cooked.pcap at http://wiki.wireshark.org/SampleCaptures under the telnet section.

Open the capture file in Wireshark and it should look like Figure 1.
NOTE: Do not do this anywhere near a production network, or any network of importance for that matter. Even if unleashing malware in a virtual environment, do it while established on a dedicated DSL connection designated for this purpose alone. You could disconnect from a network just prior to engaging the malware binary, but then you may miss some of its key behavior if it can't call home or query DNS. Please, be careful!

Here's where the fun begins. One of my favorite features in Wireshark (and Ethereal) is its ability to follow a TCP stream. In this case, if you highlight the very first packet and right-click it and select Follow TCP Stream (also available in the Analyze menu), another window will open displaying stream content from that packet's participation in a complete TCP conversation. Here's where you bring it home for your Mr. T elnet.

As you can see, Figure 2 clearly offers up the whole picture, including a username of “fake” and password of “user,” nicely color-coded to distinguish client and server.

In fact, colorization is another key feature in Wireshark. If you select the View menu and choose Coloring Rules you'll quickly determine the default color of each packet type, and you can customize as you wish, including user-contributed color filters.

A more complex scenario with Wireshark: Analyzing malware behavior

One of my favorite tasks as a security wonk is infecting a Windows XP virtual instance with a variety of malware and watching what it does on the network. While the details of the environment I prefer to do this in might be of interest to you (CentOS, VMware, Snort, OllyDbg, etc.), and it's something I'll likely write about at a later date, we'll focus here on just Wireshark's role in the process.

Using VMware Server as an example, set the capture interface to VMnet8 to gather traffic from your infected guest OS. Go to Capture, then select your Interfaces, and then choose “capture” next to the VMnet8 interface to do so.

For this exercise, imagine you've been asked to analyze an executable called bot.exe that a number of users have received in email. On your unpatched, AV-less virtual guest execute bot.exe while your Wireshark capture is running on the host OS.

It won't take much time, 60 seconds at most while connected to the Internet, to find out what bot.exe might generate in the way of network traffic.

Then stop the capture and wait for it to load in the main view. Again, you can follow along as you read by grabbing a copy of the very pcap file we're discussing from here: http://holisticinfosec.org/toolsmith/files/nov2k6/toolsmith.pcap.

In the first 16 packets, much is revealed. In #14 we see a DNS query to a strange domain name immediately followed by a response from 84.244.1.30. A quick WHOIS search reveals that this IP is in Amsterdam. The very next line really starts to clue you in. We see the local host making a call to 84.244.1.30 with a destination port of 5050. Google “port 5050” and we see a UDP reference (irrelevant, as your capture indicates TCP); the fact that Yahoo! Messenger might use TCP 5050 (interesting); but best of all, at the ISS site we learn that TCP 5050 is used by eggdrop, indicated as “the most popular bot.”

Now we're getting to the good stuff.

Let's revisit the Follow TCP Stream analysis method. Right-click on “packet 16” and choose Follow TCP Stream.

No question, your virtual guest is now a bot and talking to a C&C (command and control) server.

In the first 16 packets, much is revealed. In #14 we see a DNS query to a strange domain name immediately followed by a response from 84.244.1.30. A quick WHOIS search reveals that this IP is in Amsterdam. The very next line really starts to clue you in. We see the local host making a call to 84.244.1.30 with a destination port of 5050. Google “port 5050” and we see a UDP reference (irrelevant, as your capture indicates TCP); the fact that Yahoo! Messenger might use TCP 5050 (interesting); but best of all, at the ISS site we learn that TCP 5050 is used by eggdrop, indicated as “the most popular bot.”

Now we're getting to the good stuff.

Let's revisit the Follow TCP Stream analysis method. Right-click on “packet 16” and choose Follow TCP Stream.

No question, your virtual guest is now a bot and talking to a C&C (command and control) server. But we still don't know exactly what bot we really have. Hit Clear on your Filter toolbar after you're done...
with the TCP Stream analysis to return to the complete conversation.

We’ve learned a lot in just 16 packets, but what else might we find?
You’ll start to see some HTTP GET requests as you scroll by packet 42, as well as more interesting DNS requests. Packet 95 really got it for me: a DNS request to www.kinchan.net – not good. Packet 111 gives it all away. Right-click that packet, Follow TCP Stream again, and you’ll immediately see:

If you Google http://www.kinchan.net/cgi-bin/proxy.cgi you’ll get a fairly immediate hit on W32/Tilebot-FV. Further research at the Sophos site quickly reveals that an alias for our little friend is W32/Sdbot.

Congratulations, you’re the proud owner of an Sdbot variant, one of many polluting the Internet with ill intent.

Now let’s do something about it to protect our users...

**Firewall rules with Wireshark**

Amongst the plethora of functionality Wireshark includes is the ability to create firewall rules from a capture. Continuing on with our Sdbot pcap, highlight packet 17, choose Analyze and then Firewall ACL Rules.

You’re in immediate luck if you’re a Cisco shop. Choose Cisco IOS (extended) and you’ll see:

There are a number of other options including IPFilter, ipfw, iptables, and even a Windows Firewall option. While it’s typically not recommended to block lots of single IPs on your router (CPU utilization), you get the idea.

**In closing**

We have only touched on some very basic uses for Wireshark, particularly from an information security perspective, but nonetheless you can see the value of this application. It is, undoubtedly, one of the best open-
source tools you could ever ask for in your endeavor to protect your user base and understand your network.

As always when using tools like Wireshark, remember – have permission, be diligent, and be true to our cause.

See you next month.

---

**Figure 6. Wireshark Firewall ACL Rules**

Working out the X11 issue for Wireshark on Mac OS X:
http://www.mail-archive.com/wireshark-users@wireshark.org/msg00356.html

After you’ve worked out X11 and you need to establish interfaces:

---

**About the Author**

Russ McRee, GCIH, is a security analyst working in the Seattle area. He is a member of ISSA, Paccio, InfraGard, and CCSA (Cyber Conflict Studies Association). Russ maintains holisticinfosec.org. Contact him at russ@holisticinfosec.org.

---

**References**